```
**********************************************************
   18553 Thu Dec 26 13:48:07 2013
new/exception_lists/copyright
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2011 Nexenta Systems, Inc. All rights reserved.
# Copyright (c) 2011 by Delphix. All rights reserved.
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

syntax: glob
exception_lists/closed-bins
exception_lists/copyright
exception_lists/cstyle
exception_lists/hdrchk
usr/src/cmd/dtrace/test/tst/common/*/*.out
usr/src/cmd/krb5/kadmin/cli/kadmin_ct.c
usr/src/cmd/krb5/kadmin/cli/kadmin.h
usr/src/cmd/krb5/kadmin/cli/ss_wrapper.c
usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
usr/src/cmd/krb5/kadmin/dbutil/ovload.c
usr/src/cmd/krb5/kadmin/dbutil/string_table.c
usr/src/cmd/krb5/kadmin/dbutil/strtok.c
usr/src/cmd/krb5/kadmin/dbutil/util.c
usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
usr/src/cmd/krb5/kadmin/server/server_glue_v1.c
usr/src/cmd/krb5/krb5kdc/extern.c
usr/src/cmd/krb5/krb5kdc/kdc_util.c
usr/src/cmd/krb5/krb5kdc/policy.c
usr/src/cmd/krb5/krb5kdc/replay.c
usr/src/cmd/krb5/krb5kdc/sock2p.c
usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.c
usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
usr/src/cmd/smbsrv/smbd/eventlog.dll
usr/src/cmd/terminfo/termcap.src
usr/src/cmd/terminfo/terminfo.src
usr/src/common/acpica
usr/src/common/bzip2/LICENSE
usr/src/common/bzip2/Solaris.README.txt
usr/src/common/bzip2/bzlib.h
```



```
usr/src/common/bzip2/crctable.c
usr/src/common/bzip2/randtable.c
usr/src/common/bzip2/blocksort.c
usr/src/common/bzip2/compress.c
usr/src/common/bzip2/bzlib.c
usr/src/common/bzip2/decompress.c
usr/src/common/bzip2/bzlib_private.h
usr/src/common/bzip2/huffman.c
usr/src/common/openssl/crypto/krb5/krb5_asn.c
usr/src/common/openssl/crypto/krb5/krb5_asn.h
usr/src/grub/grub-0.97/stage2/Makefile.am
usr/src/lib/gss_mechs/mech_krb5/crypto/cksumtype_to_string.c
usr/src/lib/gss_mechs/mech_krb5/crypto/coll_proof_cksum.c
usr/src/lib/gss_mechs/mech_krb5/crypto/enctype_compare.c
usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_checksum_types.c
usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_cksum.c
usr/src/lib/gss_mechs/mech_krb5/crypto/keylengths.c
usr/src/lib/gss_mechs/mech_krb5/crypto/old/des_stringtokey.c
usr/src/lib/gss_mechs/mech_krb5/crypto/random_to_key.c
usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_cksumtype.c
usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_enctype.c
usr/src/lib/gss_mechs/mech_krb5/crypto/valid_cksumtype.c
usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
usr/src/lib/gss_mechs/mech_krb5/et/internal.h
usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
usr/src/lib/gss_mechs/mech_krb5/include/cache-addrinfo.h
usr/src/lib/gss_mechs/mech_krb5/include/cm.h
usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
usr/src/lib/gss_mechs/mech_krb5/include/db.h
usr/src/lib/gss_mechs/mech_krb5/include/fake-addrinfo.h
usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_decode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_encode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/ldap_key_seq.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cccopy.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefops.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccfns.c
```

```
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ser_cc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktadd.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfns.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktremove.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/read_servi.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_comp.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_order.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_srch.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/appdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_pr_ext.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_princ.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/chk_trans.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_addrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_creds.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_data.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_tick.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decode_kdc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decrypt_tk.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/deltat.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/enc_helper.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encode_kdc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encrypt_tk.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/free_rtree.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_via_tkt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_subkey.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_opt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/init_keyblock.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-proto.h
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kdc_rep_dc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kerrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_error.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_rep.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pr_to_salt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_error.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_rep.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_safe.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/tgtname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/valid_times.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/vic_opt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/walk_rtree.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/accessor.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/changepw.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnssrv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_hstrl.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_krbhs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/full_ipadr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_port.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_rname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/genaddrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/get_krbhst.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gmt_mktime.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/hostaddr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/localaddr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/lock_file.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/mk_faddr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_read.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_write.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-proto.h
```



```
usr/src/lib/gss_mechs/mech_krb5/krb5/os/osconfig.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/port2ip.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/prompter.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/promptusr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_msg.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_pwd.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_dom.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_iter.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/thread_safe.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/unlck_file.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/ustime.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/write_msg.c
usr/src/lib/gss_mechs/mech_krb5/krb5/posix/daemon.c
usr/src/lib/gss_mechs/mech_krb5/krb5/posix/setenv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_conv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_none.c
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcfns.c
usr/src/lib/gss_mechs/mech_krb5/mech/add_cred.c
usr/src/lib/gss_mechs/mech_krb5/mech/compare_name.c
usr/src/lib/gss_mechs/mech_krb5/mech/context_time.c
usr/src/lib/gss_mechs/mech_krb5/mech/copy_ccache.c
usr/src/lib/gss_mechs/mech_krb5/mech/disp_com_err_status.c
usr/src/lib/gss_mechs/mech_krb5/mech/disp_major_status.c
usr/src/lib/gss_mechs/mech_krb5/mech/disp_name.c
usr/src/lib/gss_mechs/mech_krb5/mech/disp_status.c
usr/src/lib/gss_mechs/mech_krb5/mech/export_name.c
usr/src/lib/gss_mechs/mech_krb5/mech/export_sec_context.c
usr/src/lib/gss_mechs/mech_krb5/mech/gss_libinit.h
usr/src/lib/gss_mechs/mech_krb5/mech/import_name.c
usr/src/lib/gss_mechs/mech_krb5/mech/indicate_mechs.c
usr/src/lib/gss_mechs/mech_krb5/mech/inq_cred.c
usr/src/lib/gss_mechs/mech_krb5/mech/inq_names.c
usr/src/lib/gss_mechs/mech_krb5/mech/oid_ops.c
usr/src/lib/gss_mechs/mech_krb5/mech/process_context_token.c
usr/src/lib/gss_mechs/mech_krb5/mech/rel_cred.c
usr/src/lib/gss_mechs/mech_krb5/mech/rel_name.c
usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid.c
usr/src/lib/gss_mechs/mech_krb5/mech/set_allowable_enctypes.c
usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer.c
usr/src/lib/gss_mechs/mech_krb5/mech/util_ctxsetup.c
usr/src/lib/gss_mechs/mech_krb5/mech/util_dup.c
usr/src/lib/gss_mechs/mech_krb5/mech/utl_nohash_validate.c
usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
usr/src/lib/gss_mechs/mech_krb5/profile/prof_get.c
usr/src/lib/gss_mechs/mech_krb5/profile/prof_set.c
usr/src/lib/gss_mechs/mech_krb5/support/errors.c
usr/src/lib/gss_mechs/mech_krb5/support/fake-addrinfo.c
usr/src/lib/gss_mechs/mech_krb5/support/init-addrinfo.c
usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
usr/src/lib/gss_mechs/mech_krb5/support/threads.c
usr/src/lib/gss_mechs/mech_krb5/support/utf8.c
usr/src/lib/krb5/dyn/dyn_append.c
usr/src/lib/krb5/dyn/dyn_create.c
usr/src/lib/krb5/dyn/dyn_debug.c
usr/src/lib/krb5/dyn/dyn_delete.c
usr/src/lib/krb5/dyn/dyn_initzero.c
usr/src/lib/krb5/dyn/dyn_insert.c
usr/src/lib/krb5/dyn/dyn_paranoid.c
usr/src/lib/krb5/dyn/dyn_put.c
usr/src/lib/krb5/dyn/dyn_realloc.c
usr/src/lib/krb5/dyn/dyn_size.c
usr/src/lib/krb5/kadm5/admin_internal.h
usr/src/lib/krb5/kadm5/admin_xdr.h
```

```
usr/src/lib/krb5/kadm5/chpass_util_strings.h
usr/src/lib/krb5/kadm5/clnt/client_handle.c
usr/src/lib/krb5/kadm5/clnt/clnt_chpass_util.c
usr/src/lib/krb5/kadm5/kadm_rpc.h
usr/src/lib/krb5/kadm5/misc_free.c
usr/src/lib/krb5/kadm5/srv/server_dict.c
usr/src/lib/krb5/kadm5/srv/server_handle.c
usr/src/lib/krb5/kadm5/srv/server_misc.c
usr/src/lib/krb5/kadm5/srv/svr_iters.c
usr/src/lib/krb5/kadm5/srv/svr_misc_free.c
usr/src/lib/krb5/kadm5/srv/svr_policy.c
usr/src/lib/krb5/kadm5/srv/xdr_alloc.c
usr/src/lib/krb5/kdb/adb_err.h
usr/src/lib/krb5/kdb/kdb5.h
usr/src/lib/krb5/kdb/keytab.c
usr/src/lib/krb5/plugins/kdb/db2/adb_policy.c
usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.c
usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_close.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_conv.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_delete.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_get.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_open.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_overflow.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_page.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_put.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_search.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_split.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_utils.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/db/db.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_bigkey.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_func.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_log2.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_page.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hsearch.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_close.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_delete.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_get.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_open.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_put.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_search.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_seq.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_utils.c
usr/src/lib/krb5/plugins/kdb/db2/pol_xdr.c
usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_fetch_mkey.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_rights.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.c
```



```
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_lib.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_profile.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_srv.c
usr/src/lib/krb5/ss/copyright.h
usr/src/lib/krb5/ss/mit-sipb-copyright.h
usr/src/lib/krb5/ss/options.c
usr/src/lib/krb5/ss/std_rqs.c
usr/src/lib/krb5/ss/utils.c
usr/src/lib/librstp/common/*.[ch]
usr/src/lib/librstp/common/[CRT]*
usr/src/test/zfs-tests/tests/functional/history/*Z
usr/src/test/zfs-tests/tests/functional/history/*txt
usr/src/uts/intel/nsmb/ioc_check.ref
usr/src/uts/intel/os/splashimage.xpm
usr/src/uts/common/gssapi/mechs/krb5/crypto/block_size.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/checksum_length.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_parity.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/weak_key.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt_length.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_crc32.c
usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_auth.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_adata.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_addr.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_auth.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_cksum.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_ctx.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_key.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/os/toffset.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_seed.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_seqnum.c
usr/src/uts/common/gssapi/mechs/krb5/mech/val_cred.c
usr/src/uts/common/io/ixgbe/ixgbe_82598.c
usr/src/uts/common/io/ixgbe/ixgbe_82599.c
usr/src/uts/common/io/ixgbe/ixgbe_api.c
usr/src/uts/common/io/ixgbe/ixgbe_api.h
usr/src/uts/common/io/ixgbe/ixgbe_common.c
usr/src/uts/common/io/ixgbe/ixgbe_common.h
usr/src/uts/common/io/ixgbe/ixgbe_osdep.h
usr/src/uts/common/io/ixgbe/ixgbe_phy.c
usr/src/uts/common/io/ixgbe/ixgbe_phy.h
usr/src/uts/common/io/ixgbe/ixgbe_type.h
usr/src/uts/common/fs/zfs/THIRDPARTYLICENSE.lz4.descrip
usr/src/uts/sparc/nsmb/ioc_check.ref
```

```
**********************************************************
   34946 Thu Dec 26 13:48:07 2013
new/exception_lists/cstyle
PANKOVs restructure
**********************************************************
```

```
  1 usr/src/cmd/krb5/kadmin/cli/kadmin_ct.c
  2 usr/src/cmd/krb5/kadmin/cli/kadmin.c
  3 usr/src/cmd/krb5/kadmin/cli/kadmin.h
  4 usr/src/cmd/krb5/kadmin/cli/keytab.c
  5 usr/src/cmd/krb5/kadmin/cli/ss_wrapper.c
  6 usr/src/cmd/krb5/kadmin/dbutil/dump.c
  7 usr/src/cmd/krb5/kadmin/dbutil/import_err.h
  8 usr/src/cmd/krb5/kadmin/dbutil/kadm5_create.c
  9 usr/src/cmd/krb5/kadmin/dbutil/kdb5_create.c
 10 usr/src/cmd/krb5/kadmin/dbutil/kdb5_destroy.c
 11 usr/src/cmd/krb5/kadmin/dbutil/kdb5_stash.c
 12 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.c
 13 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.h
 14 usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
 15 usr/src/cmd/krb5/kadmin/dbutil/ovload.c
 16 usr/src/cmd/krb5/kadmin/dbutil/string_table.c
 17 usr/src/cmd/krb5/kadmin/dbutil/string_table.h
 18 usr/src/cmd/krb5/kadmin/dbutil/strtok.c
 19 usr/src/cmd/krb5/kadmin/dbutil/util.c
 20 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
 21 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.c
 22 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
 23 usr/src/cmd/krb5/kadmin/kpasswd/tty_kpasswd.c
 24 usr/src/cmd/krb5/kadmin/ktutil/ktutil_ct.c
 25 usr/src/cmd/krb5/kadmin/ktutil/ktutil_funcs.c
 26 usr/src/cmd/krb5/kadmin/ktutil/ktutil.c
 27 usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
 28 usr/src/cmd/krb5/kadmin/server/kadm_rpc_svc.c
 29 usr/src/cmd/krb5/kadmin/server/misc.c
 30 usr/src/cmd/krb5/kadmin/server/misc.h
 31 usr/src/cmd/krb5/kadmin/server/ovsec_kadmd.c
 32 usr/src/cmd/krb5/kadmin/server/server_glue_v1.c
 33 usr/src/cmd/krb5/kadmin/server/server_stubs.c
 34 usr/src/cmd/krb5/kdestroy/kdestroy.c
 35 usr/src/cmd/krb5/kinit/kinit.c
 36 usr/src/cmd/krb5/klist/klist.c
 37 usr/src/cmd/krb5/krb5kdc/dispatch.c
 38 usr/src/cmd/krb5/krb5kdc/do_as_req.c
 39 usr/src/cmd/krb5/krb5kdc/do_tgs_req.c
 40 usr/src/cmd/krb5/krb5kdc/extern.c
 41 usr/src/cmd/krb5/krb5kdc/extern.h
 42 usr/src/cmd/krb5/krb5kdc/kdc_preauth.c
 43 usr/src/cmd/krb5/krb5kdc/kdc_util.c
 44 usr/src/cmd/krb5/krb5kdc/kdc_util.h
 45 usr/src/cmd/krb5/krb5kdc/main.c
 46 usr/src/cmd/krb5/krb5kdc/network.c
 47 usr/src/cmd/krb5/krb5kdc/policy.c
 48 usr/src/cmd/krb5/krb5kdc/policy.h
 49 usr/src/cmd/krb5/krb5kdc/replay.c
 50 usr/src/cmd/krb5/krb5kdc/sock2p.c
 51 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.c
 52 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
 53 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.c
 54 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
 55 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.c
 56 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
 57 usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.c
 58 usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
 59 usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.c
 60 usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.h
 61 usr/src/cmd/krb5/slave/kprop.c
```



```
 62 usr/src/cmd/krb5/slave/kprop.h
 63 usr/src/cmd/krb5/slave/kpropd.c
 64 usr/src/common/bzip2/bzlib.h
 65 usr/src/common/bzip2/crctable.c
 66 usr/src/common/bzip2/randtable.c
 67 usr/src/common/bzip2/blocksort.c
 68 usr/src/common/bzip2/compress.c
 69 usr/src/common/bzip2/bzlib.c
 70 usr/src/common/bzip2/decompress.c
 71 usr/src/common/bzip2/bzlib_private.h
 72 usr/src/common/bzip2/huffman.c
 73 usr/src/common/openssl/crypto/krb5/krb5_asn.c
 74 usr/src/common/openssl/crypto/krb5/krb5_asn.h
 75 usr/src/lib/gss_mechs/mech_krb5/crypto/aes/aes_s2k.c
 76 usr/src/lib/gss_mechs/mech_krb5/crypto/cksumtype_to_string.c
 77 usr/src/lib/gss_mechs/mech_krb5/crypto/coll_proof_cksum.c
 78 usr/src/lib/gss_mechs/mech_krb5/crypto/crc32/crc.c
 79 usr/src/lib/gss_mechs/mech_krb5/crypto/des/afsstring2key.c
 80 usr/src/lib/gss_mechs/mech_krb5/crypto/des/string2key.c
 81 usr/src/lib/gss_mechs/mech_krb5/crypto/dk/stringtokey.c
 82 usr/src/lib/gss_mechs/mech_krb5/crypto/enctype_compare.c
 83 usr/src/lib/gss_mechs/mech_krb5/crypto/enctype_to_string.c
 84 usr/src/lib/gss_mechs/mech_krb5/crypto/hash_provider/hash_md5.c
 85 usr/src/lib/gss_mechs/mech_krb5/crypto/hash_provider/hash_sha1.c
 86 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_checksum_types.c
 87 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_cksum.c
 88 usr/src/lib/gss_mechs/mech_krb5/crypto/keyhash_provider/hmac_md5.c
 89 usr/src/lib/gss_mechs/mech_krb5/crypto/keyhash_provider/k5_md5des.c
 90 usr/src/lib/gss_mechs/mech_krb5/crypto/keylengths.c
 91 usr/src/lib/gss_mechs/mech_krb5/crypto/make_random_key.c
 92 usr/src/lib/gss_mechs/mech_krb5/crypto/md4/md4.c
 93 usr/src/lib/gss_mechs/mech_krb5/crypto/old_api_glue.c
 94 usr/src/lib/gss_mechs/mech_krb5/crypto/old/des_stringtokey.c
 95 usr/src/lib/gss_mechs/mech_krb5/crypto/pbkdf2.c
 96 usr/src/lib/gss_mechs/mech_krb5/crypto/random_to_key.c
 97 usr/src/lib/gss_mechs/mech_krb5/crypto/state.c
 98 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_cksumtype.c
 99 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_enctype.c
100 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_key.c
101 usr/src/lib/gss_mechs/mech_krb5/crypto/valid_cksumtype.c
102 usr/src/lib/gss_mechs/mech_krb5/crypto/valid_enctype.c
103 usr/src/lib/gss_mechs/mech_krb5/et/com_err.c
104 usr/src/lib/gss_mechs/mech_krb5/et/error_message.c
105 usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
106 usr/src/lib/gss_mechs/mech_krb5/et/internal.h
107 usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
108 usr/src/lib/gss_mechs/mech_krb5/include/cache-addrinfo.h
109 usr/src/lib/gss_mechs/mech_krb5/include/cm.h
110 usr/src/lib/gss_mechs/mech_krb5/include/com_err.h
111 usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
112 usr/src/lib/gss_mechs/mech_krb5/include/db.h
113 usr/src/lib/gss_mechs/mech_krb5/include/fake-addrinfo.h
114 usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
115 usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
116 usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
117 usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
118 usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
119 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
120 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
121 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
122 usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
123 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
124 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
125 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb_dbc.h
126 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb.h
127 usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
```

```
usr/src/lib/gss_mechs/mech_krb5/include/osconf.h
usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
usr/src/lib/gss_mechs/mech_krb5/include/socket-utils.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_decode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_encode.c
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/ldap_key_seq.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_file.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_memory.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_retr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccbase.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cccopy.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefops.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccfns.c
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ser_cc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/file/ktfile.h
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_file.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_srvtab.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktadd.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktbase.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfns.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfr_entry.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktremove.c
usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/read_servi.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_comp.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_order.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_srch.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/appdefault.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_pr_ext.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_princ.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/chk_trans.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/conv_princ.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_addrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_creds.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_data.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_tick.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cp_key_cnt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decode_kdc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decrypt_tk.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/deltat.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/enc_helper.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encode_kdc.c
```



```
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encrypt_tk.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/free_rtree.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/fwd_tgt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_frm_kdc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_via_tkt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_seqnum.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_subkey.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_creds.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_in_tkt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_keytab.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_opt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_pwd.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/init_keyblock.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-proto.h
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kdc_rep_dc.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kerrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_error.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_priv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_rep.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req_ext.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_safe.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pac.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pr_to_salt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth2.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/princ_comp.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_cred.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_error.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_priv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_rep.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req_dec.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_safe.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/recvauth.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/send_tgs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/sendauth.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/set_realm.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/srv_rcache.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/str_conv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/tgtname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/valid_times.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/vic_opt.c
usr/src/lib/gss_mechs/mech_krb5/krb5/krb/walk_rtree.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/accessor.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/an_to_ln.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/ccdefname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/changepw.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/def_realm.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.h
usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnssrv.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/foreachaddr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_hstrl.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_krbhs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/full_ipadr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_port.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_rname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/genaddrs.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/get_krbhst.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/gmt_mktime.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/hostaddr.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/hst_realm.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/ktdefname.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/kuserok.c
usr/src/lib/gss_mechs/mech_krb5/krb5/os/localaddr.c
```

```
260 usr/src/lib/gss_mechs/mech_krb5/krb5/os/locate_kdc.c
261 usr/src/lib/gss_mechs/mech_krb5/krb5/os/lock_file.c
262 usr/src/lib/gss_mechs/mech_krb5/krb5/os/mk_faddr.c
263 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_read.c
264 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_write.c
265 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-proto.h
266 usr/src/lib/gss_mechs/mech_krb5/krb5/os/osconfig.c
267 usr/src/lib/gss_mechs/mech_krb5/krb5/os/port2ip.c
268 usr/src/lib/gss_mechs/mech_krb5/krb5/os/prompter.c
269 usr/src/lib/gss_mechs/mech_krb5/krb5/os/promptusr.c
270 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_msg.c
271 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_pwd.c
272 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_dom.c
273 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_iter.c
274 usr/src/lib/gss_mechs/mech_krb5/krb5/os/sendto_kdc.c
275 usr/src/lib/gss_mechs/mech_krb5/krb5/os/sn2princ.c
276 usr/src/lib/gss_mechs/mech_krb5/krb5/os/thread_safe.c
277 usr/src/lib/gss_mechs/mech_krb5/krb5/os/unlck_file.c
278 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ustime.c
279 usr/src/lib/gss_mechs/mech_krb5/krb5/os/write_msg.c
280 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/daemon.c
281 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/setenv.c
282 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
283 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_conv.c
284 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
285 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_none.c
286 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
287 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcdef.c
288 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcfns.c
289 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/ser_rc.c
290 usr/src/lib/gss_mechs/mech_krb5/mech/accept_sec_context.c
291 usr/src/lib/gss_mechs/mech_krb5/mech/acquire_cred_with_pw.c
292 usr/src/lib/gss_mechs/mech_krb5/mech/acquire_cred.c
293 usr/src/lib/gss_mechs/mech_krb5/mech/add_cred.c
294 usr/src/lib/gss_mechs/mech_krb5/mech/compare_name.c
295 usr/src/lib/gss_mechs/mech_krb5/mech/context_time.c
296 usr/src/lib/gss_mechs/mech_krb5/mech/copy_ccache.c
297 usr/src/lib/gss_mechs/mech_krb5/mech/disp_com_err_status.c
298 usr/src/lib/gss_mechs/mech_krb5/mech/disp_major_status.c
299 usr/src/lib/gss_mechs/mech_krb5/mech/disp_name.c
300 usr/src/lib/gss_mechs/mech_krb5/mech/disp_status.c
301 usr/src/lib/gss_mechs/mech_krb5/mech/export_name.c
302 usr/src/lib/gss_mechs/mech_krb5/mech/export_sec_context.c
303 usr/src/lib/gss_mechs/mech_krb5/mech/get_tkt_flags.c
304 usr/src/lib/gss_mechs/mech_krb5/mech/gss_libinit.h
305 usr/src/lib/gss_mechs/mech_krb5/mech/import_name.c
306 usr/src/lib/gss_mechs/mech_krb5/mech/indicate_mechs.c
307 usr/src/lib/gss_mechs/mech_krb5/mech/init_sec_context.c
308 usr/src/lib/gss_mechs/mech_krb5/mech/inq_context.c
309 usr/src/lib/gss_mechs/mech_krb5/mech/inq_cred.c
310 usr/src/lib/gss_mechs/mech_krb5/mech/inq_names.c
311 usr/src/lib/gss_mechs/mech_krb5/mech/krb5_gss_glue.c
312 usr/src/lib/gss_mechs/mech_krb5/mech/lucid_context.c
313 usr/src/lib/gss_mechs/mech_krb5/mech/oid_ops.c
314 usr/src/lib/gss_mechs/mech_krb5/mech/process_context_token.c
315 usr/src/lib/gss_mechs/mech_krb5/mech/rel_buffer.c
316 usr/src/lib/gss_mechs/mech_krb5/mech/rel_cred.c
317 usr/src/lib/gss_mechs/mech_krb5/mech/rel_name.c
318 usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid_set.c
319 usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid.c
320 usr/src/lib/gss_mechs/mech_krb5/mech/set_allowable_enctypes.c
321 usr/src/lib/gss_mechs/mech_krb5/mech/set_ccache.c
322 usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer_set.c
323 usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer.c
324 usr/src/lib/gss_mechs/mech_krb5/mech/util_cksum.c
325 usr/src/lib/gss_mechs/mech_krb5/mech/util_ctxsetup.c
```



```
326 usr/src/lib/gss_mechs/mech_krb5/mech/util_dup.c
327 usr/src/lib/gss_mechs/mech_krb5/mech/util_localhost.c
328 usr/src/lib/gss_mechs/mech_krb5/mech/utl_nohash_validate.c
329 usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
330 usr/src/lib/gss_mechs/mech_krb5/profile/prof_get.c
331 usr/src/lib/gss_mechs/mech_krb5/profile/prof_set.c
332 usr/src/lib/gss_mechs/mech_krb5/support/errors.c
333 usr/src/lib/gss_mechs/mech_krb5/support/fake-addrinfo.c
334 usr/src/lib/gss_mechs/mech_krb5/support/init-addrinfo.c
335 usr/src/lib/gss_mechs/mech_krb5/support/plugins.c
336 usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
337 usr/src/lib/gss_mechs/mech_krb5/support/threads.c
338 usr/src/lib/gss_mechs/mech_krb5/support/utf8_conv.c
339 usr/src/lib/gss_mechs/mech_krb5/support/utf8.c
340 usr/src/lib/krb5/dyn/dyn_append.c
341 usr/src/lib/krb5/dyn/dyn_create.c
342 usr/src/lib/krb5/dyn/dyn_debug.c
343 usr/src/lib/krb5/dyn/dyn_delete.c
344 usr/src/lib/krb5/dyn/dyn_initzero.c
345 usr/src/lib/krb5/dyn/dyn_insert.c
346 usr/src/lib/krb5/dyn/dyn_paranoid.c
347 usr/src/lib/krb5/dyn/dyn_put.c
348 usr/src/lib/krb5/dyn/dyn_realloc.c
349 usr/src/lib/krb5/dyn/dyn_size.c
350 usr/src/lib/krb5/kadm5/admin_internal.h
351 usr/src/lib/krb5/kadm5/admin_xdr.h
352 usr/src/lib/krb5/kadm5/admin.h
353 usr/src/lib/krb5/kadm5/alt_prof.c
354 usr/src/lib/krb5/kadm5/chpass_util_strings.h
355 usr/src/lib/krb5/kadm5/chpass_util.c
356 usr/src/lib/krb5/kadm5/clnt/changepw.c
357 usr/src/lib/krb5/kadm5/clnt/client_handle.c
358 usr/src/lib/krb5/kadm5/clnt/client_init.c
359 usr/src/lib/krb5/kadm5/clnt/client_internal.h
360 usr/src/lib/krb5/kadm5/clnt/client_principal.c
361 usr/src/lib/krb5/kadm5/clnt/client_rpc.c
362 usr/src/lib/krb5/kadm5/clnt/clnt_chpass_util.c
363 usr/src/lib/krb5/kadm5/clnt/clnt_policy.c
364 usr/src/lib/krb5/kadm5/clnt/clnt_privs.c
365 usr/src/lib/krb5/kadm5/clnt/logger.c
366 usr/src/lib/krb5/kadm5/kadm_err.h
367 usr/src/lib/krb5/kadm5/kadm_rpc_xdr.c
368 usr/src/lib/krb5/kadm5/kadm_rpc.h
369 usr/src/lib/krb5/kadm5/misc_free.c
370 usr/src/lib/krb5/kadm5/server_internal.h
371 usr/src/lib/krb5/kadm5/srv/adb_xdr.c
372 usr/src/lib/krb5/kadm5/srv/chgpwd.c
373 usr/src/lib/krb5/kadm5/srv/logger.c
374 usr/src/lib/krb5/kadm5/srv/server_acl.c
375 usr/src/lib/krb5/kadm5/srv/server_acl.h
376 usr/src/lib/krb5/kadm5/srv/server_dict.c
377 usr/src/lib/krb5/kadm5/srv/server_handle.c
378 usr/src/lib/krb5/kadm5/srv/server_init.c
379 usr/src/lib/krb5/kadm5/srv/server_kdb.c
380 usr/src/lib/krb5/kadm5/srv/server_misc.c
381 usr/src/lib/krb5/kadm5/srv/svr_chpass_util.c
382 usr/src/lib/krb5/kadm5/srv/svr_iters.c
383 usr/src/lib/krb5/kadm5/srv/svr_misc_free.c
384 usr/src/lib/krb5/kadm5/srv/svr_policy.c
385 usr/src/lib/krb5/kadm5/srv/svr_principal.c
386 usr/src/lib/krb5/kadm5/srv/xdr_alloc.c
387 usr/src/lib/krb5/kadm5/str_conv.c
388 usr/src/lib/krb5/kdb/adb_err.h
389 usr/src/lib/krb5/kdb/decrypt_key.c
390 usr/src/lib/krb5/kdb/encrypt_key.c
391 usr/src/lib/krb5/kdb/kdb_cpw.c
```

usr/src/lib/krb5/kdb/kdb_default.c
usr/src/lib/krb5/kdb/kdb5.c
usr/src/lib/krb5/kdb/kdb5.h
usr/src/lib/krb5/kdb/keytab.c
usr/src/lib/krb5/plugins/kdb/db2/adb_openclose.c
usr/src/lib/krb5/plugins/kdb/db2/adb_policy.c
usr/src/lib/krb5/plugins/kdb/db2/db2_exp.c
usr/src/lib/krb5/plugins/kdb/db2/kdb_compat.h
usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.c
usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.c
usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_close.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_conv.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_debug.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_delete.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_get.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_open.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_overflow.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_page.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_put.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_search.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_seq.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_split.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_utils.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/btree.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/extern.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/db/db.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/dbm.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/extern.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_bigkey.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_func.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_log2.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_page.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hsearch.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/page.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/search.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-int.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-queue.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/extern.h
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_close.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_delete.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_get.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_open.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_put.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_search.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_seq.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_utils.c
usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/recno.h
usr/src/lib/krb5/plugins/kdb/db2/pol_xdr.c
usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
usr/src/lib/krb5/plugins/kdb/ldap/ldap_exp.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap_conn.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_create.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_fetch_mkey.c

usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_main.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal2.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_rights.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.c
usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_clnt.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto.h
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_identity.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_lib.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_matching.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_profile.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_srv.c
usr/src/lib/krb5/plugins/preauth/pkinit/pkinit.h
usr/src/lib/krb5/ss/copyright.h
usr/src/lib/krb5/ss/data.c
usr/src/lib/krb5/ss/error.c
usr/src/lib/krb5/ss/execute_cmd.c
usr/src/lib/krb5/ss/help.c
usr/src/lib/krb5/ss/invocation.c
usr/src/lib/krb5/ss/list_rqs.c
usr/src/lib/krb5/ss/listen.c
usr/src/lib/krb5/ss/mit-sipb-copyright.h
usr/src/lib/krb5/ss/mk_cmds.c
usr/src/lib/krb5/ss/options.c
usr/src/lib/krb5/ss/pager.c
usr/src/lib/krb5/ss/parse.c
usr/src/lib/krb5/ss/prompt.c
usr/src/lib/krb5/ss/request_tbl.c
usr/src/lib/krb5/ss/requests.c
usr/src/lib/krb5/ss/ss_internal.h
usr/src/lib/krb5/ss/ss.h
usr/src/lib/krb5/ss/std_rqs.c
usr/src/lib/krb5/ss/utils.c
usr/src/lib/libgss/g_glue.c
usr/src/lib/librstp/common/base.h
usr/src/lib/librstp/common/choose.h
usr/src/lib/librstp/common/edge.c
usr/src/lib/librstp/common/edge.h
usr/src/lib/librstp/common/migrate.c
usr/src/lib/librstp/common/migrate.h
usr/src/lib/librstp/common/p2p.c
usr/src/lib/librstp/common/p2p.h
usr/src/lib/librstp/common/pcost.c
usr/src/lib/librstp/common/pcost.h
usr/src/lib/librstp/common/port.c
usr/src/lib/librstp/common/port.h

```
usr/src/lib/librstp/common/portinfo.c
usr/src/lib/librstp/common/portinfo.h
usr/src/lib/librstp/common/rolesel.c
usr/src/lib/librstp/common/rolesel.h
usr/src/lib/librstp/common/roletrns.c
usr/src/lib/librstp/common/roletrns.h
usr/src/lib/librstp/common/statmch.c
usr/src/lib/librstp/common/statmch.h
usr/src/lib/librstp/common/stp_bpdu.h
usr/src/lib/librstp/common/stp_in.c
usr/src/lib/librstp/common/stp_in.h
usr/src/lib/librstp/common/stp_to.h
usr/src/lib/librstp/common/stp_vectors.h
usr/src/lib/librstp/common/stpm.c
usr/src/lib/librstp/common/stpm.h
usr/src/lib/librstp/common/stpmgmt.c
usr/src/lib/librstp/common/sttrans.c
usr/src/lib/librstp/common/sttrans.h
usr/src/lib/librstp/common/times.c
usr/src/lib/librstp/common/times.h
usr/src/lib/librstp/common/topoch.c
usr/src/lib/librstp/common/topoch.h
usr/src/lib/librstp/common/transmit.c
usr/src/lib/librstp/common/transmit.h
usr/src/lib/librstp/common/uid_stp.h
usr/src/lib/librstp/common/vector.c
usr/src/lib/librstp/common/vector.h
usr/src/uts/common/gssapi/gssapi.h
usr/src/uts/common/gssapi/mechs/krb5/crypto/block_size.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/checksum_length.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/cksumtypes.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/combine_keys.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/crc32/crc32.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/decrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/default_state.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/d3_cbc.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_cbc.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_parity.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/des/weak_key.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/checksum.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/derive.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_decrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_encrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/arcfour_provider.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des3.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt_length.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/etypes.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_crc32.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_kmd5.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_ksha1.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/hmac.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/descbc.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k_hmac_md5.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k5_kmd5des.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/make_checksum.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/mandatory_sumtype.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/nfold.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_decrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_encrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/prng.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_decrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_encrypt.c
usr/src/uts/common/gssapi/mechs/krb5/crypto/verify_checksum.c
usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
```



```
usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/old.h
usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_athctr.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_auth.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_cksum.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_key.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_princ.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/init_ctx.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/kfree.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/parse.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_actx.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_adata.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_addr.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_auth.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_cksum.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_ctx.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_key.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_princ.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/serialize.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/unparse.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/os/c_ustime.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/os/init_os_ctx.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/os/timeofday.c
usr/src/uts/common/gssapi/mechs/krb5/krb5/os/toffset.c
usr/src/uts/common/gssapi/mechs/krb5/mech/delete_sec_context.c
usr/src/uts/common/gssapi/mechs/krb5/mech/gssapi_krb5.c
usr/src/uts/common/gssapi/mechs/krb5/mech/import_sec_context.c
usr/src/uts/common/gssapi/mechs/krb5/mech/k5seal.c
usr/src/uts/common/gssapi/mechs/krb5/mech/k5sealv3.c
usr/src/uts/common/gssapi/mechs/krb5/mech/k5unseal.c
usr/src/uts/common/gssapi/mechs/krb5/mech/seal.c
usr/src/uts/common/gssapi/mechs/krb5/mech/ser_sctx.c
usr/src/uts/common/gssapi/mechs/krb5/mech/sign.c
usr/src/uts/common/gssapi/mechs/krb5/mech/unseal.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_crypt.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_ordering.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_seed.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_seqnum.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_set.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_token.c
usr/src/uts/common/gssapi/mechs/krb5/mech/util_validate.c
usr/src/uts/common/gssapi/mechs/krb5/mech/val_cred.c
```

usr/src/uts/common/gssapi/mechs/krb5/mech/verify.c
usr/src/uts/common/gssapi/mechs/krb5/mech/wrap_size_limit.c
usr/src/uts/common/io/e1000api/e1000_80003es2lan.c
usr/src/uts/common/io/e1000api/e1000_80003es2lan.h
usr/src/uts/common/io/e1000api/e1000_82540.c
usr/src/uts/common/io/e1000api/e1000_82541.c
usr/src/uts/common/io/e1000api/e1000_82541.h
usr/src/uts/common/io/e1000api/e1000_82542.c
usr/src/uts/common/io/e1000api/e1000_82543.c
usr/src/uts/common/io/e1000api/e1000_82543.h
usr/src/uts/common/io/e1000api/e1000_82571.c
usr/src/uts/common/io/e1000api/e1000_82571.h
usr/src/uts/common/io/e1000api/e1000_82575.c
usr/src/uts/common/io/e1000api/e1000_82575.h
usr/src/uts/common/io/e1000api/e1000_api.c
usr/src/uts/common/io/e1000api/e1000_api.h
usr/src/uts/common/io/e1000api/e1000_defines.h
usr/src/uts/common/io/e1000api/e1000_hw.h
usr/src/uts/common/io/e1000api/e1000_i210.c
usr/src/uts/common/io/e1000api/e1000_i210.h
usr/src/uts/common/io/e1000api/e1000_ich8lan.c
usr/src/uts/common/io/e1000api/e1000_ich8lan.h
usr/src/uts/common/io/e1000api/e1000_mac.c
usr/src/uts/common/io/e1000api/e1000_mac.h
usr/src/uts/common/io/e1000api/e1000_manage.c
usr/src/uts/common/io/e1000api/e1000_manage.h
usr/src/uts/common/io/e1000api/e1000_mbx.c
usr/src/uts/common/io/e1000api/e1000_mbx.h
usr/src/uts/common/io/e1000api/e1000_nvm.c
usr/src/uts/common/io/e1000api/e1000_nvm.h
usr/src/uts/common/io/e1000api/e1000_phy.c
usr/src/uts/common/io/e1000api/e1000_phy.h
usr/src/uts/common/io/e1000api/e1000_regs.h
usr/src/uts/common/io/e1000api/e1000_vf.c
usr/src/uts/common/io/e1000api/e1000_vf.h
usr/src/uts/common/io/ixgbe/ixgbe_82598.c
usr/src/uts/common/io/ixgbe/ixgbe_82598.h
usr/src/uts/common/io/ixgbe/ixgbe_82599.c
usr/src/uts/common/io/ixgbe/ixgbe_82599.h
usr/src/uts/common/io/ixgbe/ixgbe_api.c
usr/src/uts/common/io/ixgbe/ixgbe_api.h
usr/src/uts/common/io/ixgbe/ixgbe_common.c
usr/src/uts/common/io/ixgbe/ixgbe_common.h
usr/src/uts/common/io/ixgbe/ixgbe_mbx.c
usr/src/uts/common/io/ixgbe/ixgbe_mbx.h
usr/src/uts/common/io/ixgbe/ixgbe_osdep.h
usr/src/uts/common/io/ixgbe/ixgbe_phy.c
usr/src/uts/common/io/ixgbe/ixgbe_phy.h
usr/src/uts/common/io/ixgbe/ixgbe_type.h
usr/src/uts/common/io/ixgbe/ixgbe_x540.c
usr/src/uts/common/io/ixgbe/ixgbe_x540.h
usr/src/common/acpica
usr/src/uts/intel/io/acpica/debugger/dbcmds.c
usr/src/uts/intel/io/acpica/debugger/dbdisply.c
usr/src/uts/intel/io/acpica/debugger/dbexec.c
usr/src/uts/intel/io/acpica/debugger/dbfileio.c
usr/src/uts/intel/io/acpica/debugger/dbhistry.c
usr/src/uts/intel/io/acpica/debugger/dbinput.c
usr/src/uts/intel/io/acpica/debugger/dbmethod.c
usr/src/uts/intel/io/acpica/debugger/dbnames.c
usr/src/uts/intel/io/acpica/debugger/dbstats.c
usr/src/uts/intel/io/acpica/debugger/dbutils.c
usr/src/uts/intel/io/acpica/debugger/dbxface.c
usr/src/uts/intel/io/acpica/disassembler/dmbuffer.c
usr/src/uts/intel/io/acpica/disassembler/dmnames.c
usr/src/uts/intel/io/acpica/disassembler/dmobject.c

usr/src/uts/intel/io/acpica/disassembler/dmopcode.c
usr/src/uts/intel/io/acpica/disassembler/dmresrc.c
usr/src/uts/intel/io/acpica/disassembler/dmresrcl.c
usr/src/uts/intel/io/acpica/disassembler/dmresrcs.c
usr/src/uts/intel/io/acpica/disassembler/dmutils.c
usr/src/uts/intel/io/acpica/disassembler/dmwalk.c
usr/src/uts/intel/io/acpica/dispatcher/dsargs.c
usr/src/uts/intel/io/acpica/dispatcher/dscontrol.c
usr/src/uts/intel/io/acpica/dispatcher/dsfield.c
usr/src/uts/intel/io/acpica/dispatcher/dsinit.c
usr/src/uts/intel/io/acpica/dispatcher/dsmethod.c
usr/src/uts/intel/io/acpica/dispatcher/dsmthdat.c
usr/src/uts/intel/io/acpica/dispatcher/dsobject.c
usr/src/uts/intel/io/acpica/dispatcher/dsopcode.c
usr/src/uts/intel/io/acpica/dispatcher/dsutils.c
usr/src/uts/intel/io/acpica/dispatcher/dswexec.c
usr/src/uts/intel/io/acpica/dispatcher/dswload.c
usr/src/uts/intel/io/acpica/dispatcher/dswload2.c
usr/src/uts/intel/io/acpica/dispatcher/dswscope.c
usr/src/uts/intel/io/acpica/dispatcher/dswstate.c
usr/src/uts/intel/io/acpica/events/evevent.c
usr/src/uts/intel/io/acpica/events/evglock.c
usr/src/uts/intel/io/acpica/events/evgpe.c
usr/src/uts/intel/io/acpica/events/evgpeblk.c
usr/src/uts/intel/io/acpica/events/evgpeinit.c
usr/src/uts/intel/io/acpica/events/evgpeutil.c
usr/src/uts/intel/io/acpica/events/evmisc.c
usr/src/uts/intel/io/acpica/events/evregion.c
usr/src/uts/intel/io/acpica/events/evrgnini.c
usr/src/uts/intel/io/acpica/events/evsci.c
usr/src/uts/intel/io/acpica/events/evxface.c
usr/src/uts/intel/io/acpica/events/evxfevnt.c
usr/src/uts/intel/io/acpica/events/evxfgpe.c
usr/src/uts/intel/io/acpica/events/evxfregn.c
usr/src/uts/intel/io/acpica/executer/exconfig.c
usr/src/uts/intel/io/acpica/executer/exconvrt.c
usr/src/uts/intel/io/acpica/executer/excreate.c
usr/src/uts/intel/io/acpica/executer/exdebug.c
usr/src/uts/intel/io/acpica/executer/exdump.c
usr/src/uts/intel/io/acpica/executer/exfield.c
usr/src/uts/intel/io/acpica/executer/exfldio.c
usr/src/uts/intel/io/acpica/executer/exmisc.c
usr/src/uts/intel/io/acpica/executer/exmutex.c
usr/src/uts/intel/io/acpica/executer/exnames.c
usr/src/uts/intel/io/acpica/executer/exoparg1.c
usr/src/uts/intel/io/acpica/executer/exoparg2.c
usr/src/uts/intel/io/acpica/executer/exoparg3.c
usr/src/uts/intel/io/acpica/executer/exoparg6.c
usr/src/uts/intel/io/acpica/executer/exprep.c
usr/src/uts/intel/io/acpica/executer/exregion.c
usr/src/uts/intel/io/acpica/executer/exresnte.c
usr/src/uts/intel/io/acpica/executer/exresolv.c
usr/src/uts/intel/io/acpica/executer/exresop.c
usr/src/uts/intel/io/acpica/executer/exstore.c
usr/src/uts/intel/io/acpica/executer/exstoren.c
usr/src/uts/intel/io/acpica/executer/exstorob.c
usr/src/uts/intel/io/acpica/executer/exsystem.c
usr/src/uts/intel/io/acpica/executer/exutils.c
usr/src/uts/intel/io/acpica/hardware/hwacpi.c
usr/src/uts/intel/io/acpica/hardware/hwgpe.c
usr/src/uts/intel/io/acpica/hardware/hwpci.c
usr/src/uts/intel/io/acpica/hardware/hwregs.c
usr/src/uts/intel/io/acpica/hardware/hwsleep.c
usr/src/uts/intel/io/acpica/hardware/hwtimer.c
usr/src/uts/intel/io/acpica/hardware/hwvalid.c
usr/src/uts/intel/io/acpica/hardware/hwxface.c

```
usr/src/uts/intel/io/acpica/namespace/nsaccess.c
usr/src/uts/intel/io/acpica/namespace/nsalloc.c
usr/src/uts/intel/io/acpica/namespace/nsdump.c
usr/src/uts/intel/io/acpica/namespace/nsdumpdv.c
usr/src/uts/intel/io/acpica/namespace/nseval.c
usr/src/uts/intel/io/acpica/namespace/nsinit.c
usr/src/uts/intel/io/acpica/namespace/nsload.c
usr/src/uts/intel/io/acpica/namespace/nsnames.c
usr/src/uts/intel/io/acpica/namespace/nsobject.c
usr/src/uts/intel/io/acpica/namespace/nsparse.c
usr/src/uts/intel/io/acpica/namespace/nspredef.c
usr/src/uts/intel/io/acpica/namespace/nsrepair.c
usr/src/uts/intel/io/acpica/namespace/nsrepair2.c
usr/src/uts/intel/io/acpica/namespace/nssearch.c
usr/src/uts/intel/io/acpica/namespace/nsutils.c
usr/src/uts/intel/io/acpica/namespace/nswalk.c
usr/src/uts/intel/io/acpica/namespace/nsxfeval.c
usr/src/uts/intel/io/acpica/namespace/nsxfname.c
usr/src/uts/intel/io/acpica/namespace/nsxfobj.c
usr/src/uts/intel/io/acpica/parser/psargs.c
usr/src/uts/intel/io/acpica/parser/psloop.c
usr/src/uts/intel/io/acpica/parser/psopcode.c
usr/src/uts/intel/io/acpica/parser/psparse.c
usr/src/uts/intel/io/acpica/parser/psscope.c
usr/src/uts/intel/io/acpica/parser/pstree.c
usr/src/uts/intel/io/acpica/parser/psutils.c
usr/src/uts/intel/io/acpica/parser/pswalk.c
usr/src/uts/intel/io/acpica/parser/psxface.c
usr/src/uts/intel/io/acpica/resources/rsaddr.c
usr/src/uts/intel/io/acpica/resources/rscalc.c
usr/src/uts/intel/io/acpica/resources/rscreate.c
usr/src/uts/intel/io/acpica/resources/rsdump.c
usr/src/uts/intel/io/acpica/resources/rsinfo.c
usr/src/uts/intel/io/acpica/resources/rsio.c
usr/src/uts/intel/io/acpica/resources/rsirq.c
usr/src/uts/intel/io/acpica/resources/rslist.c
usr/src/uts/intel/io/acpica/resources/rsmemory.c
usr/src/uts/intel/io/acpica/resources/rsmisc.c
usr/src/uts/intel/io/acpica/resources/rsutils.c
usr/src/uts/intel/io/acpica/resources/rsxface.c
usr/src/uts/intel/io/acpica/tables/tbfadt.c
usr/src/uts/intel/io/acpica/tables/tbfind.c
usr/src/uts/intel/io/acpica/tables/tbinstal.c
usr/src/uts/intel/io/acpica/tables/tbutils.c
usr/src/uts/intel/io/acpica/tables/tbxface.c
usr/src/uts/intel/io/acpica/tables/tbxfroot.c
usr/src/uts/intel/io/acpica/utilities/utalloc.c
usr/src/uts/intel/io/acpica/utilities/utcache.c
usr/src/uts/intel/io/acpica/utilities/utclib.c
usr/src/uts/intel/io/acpica/utilities/utcopy.c
usr/src/uts/intel/io/acpica/utilities/utdebug.c
usr/src/uts/intel/io/acpica/utilities/utdecode.c
usr/src/uts/intel/io/acpica/utilities/utdelete.c
usr/src/uts/intel/io/acpica/utilities/uteval.c
usr/src/uts/intel/io/acpica/utilities/utglobal.c
usr/src/uts/intel/io/acpica/utilities/utids.c
usr/src/uts/intel/io/acpica/utilities/utinit.c
usr/src/uts/intel/io/acpica/utilities/utlock.c
usr/src/uts/intel/io/acpica/utilities/utmath.c
usr/src/uts/intel/io/acpica/utilities/utmisc.c
usr/src/uts/intel/io/acpica/utilities/utmutex.c
usr/src/uts/intel/io/acpica/utilities/utobject.c
usr/src/uts/intel/io/acpica/utilities/utosi.c
usr/src/uts/intel/io/acpica/utilities/utresrc.c
usr/src/uts/intel/io/acpica/utilities/utstate.c
usr/src/uts/intel/io/acpica/utilities/uttrack.c
```



```
usr/src/uts/intel/io/acpica/utilities/utxface.c
usr/src/uts/intel/io/acpica/utilities/utxferror.c
usr/src/uts/intel/sys/acpi/acapps.h
usr/src/uts/intel/sys/acpi/accommon.h
usr/src/uts/intel/sys/acpi/acconfig.h
usr/src/uts/intel/sys/acpi/acdebug.h
usr/src/uts/intel/sys/acpi/acdisasm.h
usr/src/uts/intel/sys/acpi/acdispat.h
usr/src/uts/intel/sys/acpi/acevents.h
usr/src/uts/intel/sys/acpi/acexcep.h
usr/src/uts/intel/sys/acpi/acglobal.h
usr/src/uts/intel/sys/acpi/achware.h
usr/src/uts/intel/sys/acpi/acinterp.h
usr/src/uts/intel/sys/acpi/aclocal.h
usr/src/uts/intel/sys/acpi/acmacros.h
usr/src/uts/intel/sys/acpi/acnames.h
usr/src/uts/intel/sys/acpi/acnamesp.h
usr/src/uts/intel/sys/acpi/acobject.h
usr/src/uts/intel/sys/acpi/acopcode.h
usr/src/uts/intel/sys/acpi/acoutput.h
usr/src/uts/intel/sys/acpi/acparser.h
usr/src/uts/intel/sys/acpi/acpi.h
usr/src/uts/intel/sys/acpi/acpiosxf.h
usr/src/uts/intel/sys/acpi/acpixf.h
usr/src/uts/intel/sys/acpi/acpredef.h
usr/src/uts/intel/sys/acpi/acresrc.h
usr/src/uts/intel/sys/acpi/acrestyp.h
usr/src/uts/intel/sys/acpi/acstruct.h
usr/src/uts/intel/sys/acpi/actables.h
usr/src/uts/intel/sys/acpi/actbl.h
usr/src/uts/intel/sys/acpi/actbl1.h
usr/src/uts/intel/sys/acpi/actbl2.h
usr/src/uts/intel/sys/acpi/actypes.h
usr/src/uts/intel/sys/acpi/acutils.h
usr/src/uts/intel/sys/acpi/amlcode.h
usr/src/uts/intel/sys/acpi/amlresrc.h
usr/src/uts/intel/sys/acpi/platform/accygwin.h
usr/src/uts/intel/sys/acpi/platform/acefi.h
usr/src/uts/intel/sys/acpi/platform/acenv.h
usr/src/uts/intel/sys/acpi/platform/acfreebsd.h
usr/src/uts/intel/sys/acpi/platform/acgcc.h
usr/src/uts/intel/sys/acpi/platform/acintel.h
usr/src/uts/intel/sys/acpi/platform/aclinux.h
usr/src/uts/intel/sys/acpi/platform/acmsvc.h
usr/src/uts/intel/sys/acpi/platform/acnetbsd.h
usr/src/uts/intel/sys/acpi/platform/acos2.h
usr/src/uts/intel/sys/acpi/platform/acsolaris.h
usr/src/uts/intel/sys/acpi/platform/acwin.h
usr/src/uts/intel/sys/acpi/platform/acwin64.h
```

```
**********************************************************
    8480 Thu Dec 26 13:48:08 2013
new/exception_lists/hdrchk
PANKOVs restructure
**********************************************************
```

```
  1 usr/src/cmd/krb5/kadmin/cli/kadmin.h
  2 usr/src/cmd/krb5/kadmin/dbutil/import_err.h
  3 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.h
  4 usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
  5 usr/src/cmd/krb5/kadmin/dbutil/string_table.h
  6 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
  7 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
  8 usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
  9 usr/src/cmd/krb5/kadmin/server/misc.h
 10 usr/src/cmd/krb5/krb5kdc/extern.h
 11 usr/src/cmd/krb5/krb5kdc/kdc_util.h
 12 usr/src/cmd/krb5/krb5kdc/policy.h
 13 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
 14 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
 15 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
 16 usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
 17 usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.h
 18 usr/src/cmd/krb5/slave/kprop.h
 19 usr/src/cmd/localedef/localedef.h
 20 usr/src/common/openssl/crypto/krb5/krb5_asn.h
 21 usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
 22 usr/src/lib/gss_mechs/mech_krb5/et/internal.h
 23 usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
 24 usr/src/lib/gss_mechs/mech_krb5/include/cache-addrinfo.h
 25 usr/src/lib/gss_mechs/mech_krb5/include/cm.h
 26 usr/src/lib/gss_mechs/mech_krb5/include/com_err.h
 27 usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
 28 usr/src/lib/gss_mechs/mech_krb5/include/db.h
 29 usr/src/lib/gss_mechs/mech_krb5/include/fake-addrinfo.h
 30 usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
 31 usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
 32 usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
 33 usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
 34 usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
 35 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
 36 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
 37 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
 38 usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
 39 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
 40 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
 41 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb_dbc.h
 42 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb.h
 43 usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
 44 usr/src/lib/gss_mechs/mech_krb5/include/osconf.h
 45 usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
 46 usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
 47 usr/src/lib/gss_mechs/mech_krb5/include/socket-utils.h
 48 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
 49 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
 50 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
 51 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
 52 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
 53 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
 54 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
 55 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
 56 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
 57 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
 58 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
 59 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
 60 usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
 61 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/file/ktfile.h
```



```
 62 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
 63 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
 64 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-proto.h
 65 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.h
 66 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-proto.h
 67 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
 68 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
 69 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
 70 usr/src/lib/gss_mechs/mech_krb5/mech/gss_libinit.h
 71 usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
 72 usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
 73 usr/src/lib/krb5/kadm5/admin_internal.h
 74 usr/src/lib/krb5/kadm5/admin_xdr.h
 75 usr/src/lib/krb5/kadm5/admin.h
 76 usr/src/lib/krb5/kadm5/chpass_util_strings.h
 77 usr/src/lib/krb5/kadm5/clnt/client_internal.h
 78 usr/src/lib/krb5/kadm5/kadm_err.h
 79 usr/src/lib/krb5/kadm5/kadm_rpc.h
 80 usr/src/lib/krb5/kadm5/server_internal.h
 81 usr/src/lib/krb5/kadm5/srv/server_acl.h
 82 usr/src/lib/krb5/kdb/adb_err.h
 83 usr/src/lib/krb5/kdb/kdb5.h
 84 usr/src/lib/krb5/plugins/kdb/db2/kdb_compat.h
 85 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
 86 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
 87 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/btree.h
 88 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/extern.h
 89 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/extern.h
 90 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.h
 91 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/page.h
 92 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/search.h
 93 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-int.h
 94 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
 95 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-queue.h
 96 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.h
 97 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/extern.h
 98 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/recno.h
 99 usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
100 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.h
101 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
102 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
103 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
104 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
105 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_main.h
106 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.h
107 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
108 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
109 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
110 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
111 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
112 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
113 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
114 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.h
115 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto.h
116 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit.h
117 usr/src/lib/krb5/ss/copyright.h
118 usr/src/lib/krb5/ss/mit-sipb-copyright.h
119 usr/src/lib/krb5/ss/ss_internal.h
120 usr/src/lib/krb5/ss/ss.h
121 usr/src/lib/libc/port/locale/utils.h
122 usr/src/lib/librstp/common/base.h
123 usr/src/lib/librstp/common/choose.h
124 usr/src/lib/librstp/common/edge.h
125 usr/src/lib/librstp/common/migrate.h
126 usr/src/lib/librstp/common/p2p.h
127 usr/src/lib/librstp/common/pcost.h
```

```
usr/src/lib/librstp/common/port.h
usr/src/lib/librstp/common/portinfo.h
usr/src/lib/librstp/common/rolesel.h
usr/src/lib/librstp/common/roletrns.h
usr/src/lib/librstp/common/statmch.h
usr/src/lib/librstp/common/stp_bpdu.h
usr/src/lib/librstp/common/stp_in.h
usr/src/lib/librstp/common/stp_to.h
usr/src/lib/librstp/common/stp_vectors.h
usr/src/lib/librstp/common/stpm.h
usr/src/lib/librstp/common/sttrans.h
usr/src/lib/librstp/common/times.h
usr/src/lib/librstp/common/topoch.h
usr/src/lib/librstp/common/transmit.h
usr/src/lib/librstp/common/uid_stp.h
usr/src/lib/librstp/common/vector.h
usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
usr/src/uts/common/gssapi/mechs/krb5/include/old.h
usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
usr/src/uts/common/io/ixgbe/ixgbe_common.h
usr/src/common/acpica
usr/src/uts/intel/sys/acpi/acdebug.h
usr/src/uts/intel/sys/acpi/acdisasm.h
usr/src/uts/intel/sys/acpi/acevents.h
usr/src/uts/intel/sys/acpi/acinterp.h
usr/src/uts/intel/sys/acpi/acmacros.h
usr/src/uts/intel/sys/acpi/acnames.h
usr/src/uts/intel/sys/acpi/acpredef.h
usr/src/uts/intel/sys/acpi/acresrc.h
usr/src/uts/intel/sys/acpi/acstruct.h
usr/src/uts/intel/sys/acpi/amlresrc.h
usr/src/uts/intel/sys/acpi/platform/acwin64.h
```

```
**********************************************************
   35161 Thu Dec 26 13:48:08 2013
new/usr/src/Makefile.master
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

#
# Makefile.master, global definitions for system source
#
ROOT=           /proto

#
# Adjunct root, containing an additional proto area to be used for headers
# and libraries.
#
ADJUNCT_PROTO=

#
# Adjunct for building things that run on the build machine.
#
NATIVE_ADJUNCT= /usr

#
# RELEASE_BUILD should be cleared for final release builds.
# NOT_RELEASE_BUILD is exactly what the name implies.
#
# INTERNAL_RELEASE_BUILD is a subset of RELEASE_BUILD. It mostly controls
# identification strings. Enabling RELEASE_BUILD automatically enables
# INTERNAL_RELEASE_BUILD.
#
# STRIP_COMMENTS toggles comment section striping. Generally the same setting
# as INTERNAL_RELEASE_BUILD.
#
# __GNUC toggles the building of ON components using gcc and related tools.
# Normally set to '#', set it to '' to do gcc build.
#
# The declaration POUND_SIGN is always '#'. This is needed to get around the
# make feature that '#' is always a comment delimiter, even when escaped or
# quoted. We use this macro expansion method to get POUND_SIGN rather than
# always breaking out a shell because the general case can cause a noticable
```



```
# slowdown in build times when so many Makefiles include Makefile.master.
#
# While the majority of users are expected to override the setting below
# with an env file (via nightly or bldenv), if you aren't building that way
# (ie, you're using "ws" or some other bootstrapping method) then you need
# this definition in order to avoid the subshell invocation mentioned above.
#

PRE_POUND=                              pre\#
POUND_SIGN=                             $(PRE_POUND:pre\%=%)

NOT_RELEASE_BUILD=
INTERNAL_RELEASE_BUILD=                 $(POUND_SIGN)
RELEASE_BUILD=                          $(POUND_SIGN)
$(RELEASE_BUILD)NOT_RELEASE_BUILD=      $(POUND_SIGN)
$(RELEASE_BUILD)INTERNAL_RELEASE_BUILD=
PATCH_BUILD=                            $(POUND_SIGN)

# SPARC_BLD is '#' for an Intel build.
# INTEL_BLD is '#' for a Sparc build.
SPARC_BLD_1=    $(MACH:i386=$(POUND_SIGN))
SPARC_BLD=      $(SPARC_BLD_1:sparc=)
INTEL_BLD_1=    $(MACH:sparc=$(POUND_SIGN))
INTEL_BLD=      $(INTEL_BLD_1:i386=)

STRIP_COMMENTS= $(INTERNAL_RELEASE_BUILD)

# The variables below control the compilers used during the build.
# There are a number of permutations.
#
# __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
# one is not POUND_SIGN is the primary, with the other as the shadow.  They
# may also be used to control entirely compiler-specific Makefile assignments.
# __SUNC and Sun Studio are the default.
#
# __GNUC64 indicates that the 64bit build should use the GNU C compiler.
# There is no Sun C analogue.
#
# The following version-specific options are operative regardless of which
# compiler is primary, and control the versions of the given compilers to be
# used.  They also allow compiler-version specific Makefile fragments.
#

__GNUC=                 $(POUND_SIGN)
$(__GNUC)__SUNC=        $(POUND_SIGN)
__GNUC64=               $(__GNUC)

# CLOSED is the root of the tree that contains source which isn't released
# as open source
CLOSED=         $(SRC)/../closed

# BUILD_TOOLS is the root of all tools including compilers.
# ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld.

BUILD_TOOLS=            /ws/onnv-tools
ONBLD_TOOLS=            $(BUILD_TOOLS)/onbld

JAVA_ROOT=      /usr/java

SFW_ROOT=       /usr/sfw
SFWINCDIR=      $(SFW_ROOT)/include
SFWLIBDIR=      $(SFW_ROOT)/lib
SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

GCC_ROOT=       /opt/gcc/4.4.4
GCCLIBDIR=      $(GCC_ROOT)/lib
```

```
128 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

130 DOCBOOK_XSL_ROOT=       /usr/share/sgml/docbook/xsl-stylesheets

132 RPCGEN=         /usr/bin/rpcgen
133 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
134 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
135 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
136 ECHO=           echo
137 INS=            install
138 TRUE=           true
139 SYMLINK=        /usr/bin/ln -s
140 LN=             /usr/bin/ln
141 CHMOD=          /usr/bin/chmod
142 MV=             /usr/bin/mv -f
143 RM=             /usr/bin/rm -f
144 CUT=            /usr/bin/cut
145 NM=             /usr/ccs/bin/nm
146 DIFF=           /usr/bin/diff
147 GREP=           /usr/bin/grep
148 EGREP=          /usr/bin/egrep
149 ELFWRAP=        /usr/bin/elfwrap
150 KSH93=          /usr/bin/ksh93
151 SED=            /usr/bin/sed
152 NAWK=           /usr/bin/nawk
153 CP=             /usr/bin/cp -f
154 MCS=            /usr/ccs/bin/mcs
155 CAT=            /usr/bin/cat
156 ELFDUMP=        /usr/ccs/bin/elfdump
157 M4=             /usr/ccs/bin/m4
158 STRIP=          /usr/ccs/bin/strip
159 LEX=            /usr/ccs/bin/lex
160 FLEX=           /usr/bin/flex
159 FLEX=           $(SFW_ROOT)/bin/flex
161 YACC=           /usr/ccs/bin/yacc
162 BISON=          /usr/bin/bison
163 CPP=            /usr/lib/cpp
164 JAVAC=          $(JAVA_ROOT)/bin/javac
165 JAVAH=          $(JAVA_ROOT)/bin/javah
166 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
167 RMIC=           $(JAVA_ROOT)/bin/rmic
168 JAR=            $(JAVA_ROOT)/bin/jar
169 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
170 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
171 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
172 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
173 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
174 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
175 CTFCVTPTBL=     $(ONBLD_TOOLS)/bin/ctfcvtptbl
176 CTFFINDMOD=     $(ONBLD_TOOLS)/bin/ctffindmod
177 XREF=           $(ONBLD_TOOLS)/bin/xref
178 FIND=           /usr/bin/find
179 PERL=           /usr/bin/perl
180 PYTHON_26=      /usr/bin/python2.6
181 PYTHON=         $(PYTHON_26)
182 SORT=           /usr/bin/sort
183 TOUCH=          /usr/bin/touch
184 WC=             /usr/bin/wc
185 XARGS=          /usr/bin/xargs
186 ELFEDIT=        /usr/bin/elfedit
187 ELFSIGN=        /usr/bin/elfsign
188 DTRACE=         /usr/sbin/dtrace -xnolibs
189 UNIQ=           /usr/bin/uniq
190 TAR=            /usr/bin/tar
191 ASTBINDIR=      /usr/ast/bin
192 MSGCC=          $(ASTBINDIR)/msgcc
```



```
194 FILEMODE=       644
195 DIRMODE=        755

197 #
198 # The version of the patch makeup table optimized for build-time use.  Used
199 # during patch builds only.
200 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

202 # Declare that nothing should be built in parallel.
203 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
204 .NO_PARALLEL:

206 # For stylistic checks
207 #
208 # Note that the X and C checks are not used at this time and may need
209 # modification when they are actually used.
210 #
211 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
212 CSTYLE_TAIL=
213 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
214 HDRCHK_TAIL=
215 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

217 DOT_H_CHECK=    \
218         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
219         $(HDRCHK) $< $(HDRCHK_TAIL)

221 DOT_X_CHECK=    \
222         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
223         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

225 DOT_C_CHECK=    \
226         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

228 MANIFEST_CHECK= \
229         @$(ECHO) "checking $<"; \
230         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
231         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
232         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
233         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

235 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
236 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
237 # installs and renames at once
238 #
239 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

241 # install a link
242 INSLINKTARGET=  $<
243 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
244 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

246 #
247 # Python bakes the mtime of the .py file into the compiled .pyc and
248 # rebuilds if the baked-in mtime != the mtime of the source file
249 # (rather than only if it's less than), thus when installing python
250 # files we must make certain to not adjust the mtime of the source
251 # (.py) file.
252 #
253 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

255 # MACH must be set in the shell environment per uname -p on the build host
256 # More specific architecture variables should be set in lower makefiles.
257 #
258 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
```

```
259 # architectures on which we do not build 64-bit versions.
260 # (There are no such architectures at the moment.)
261 #
262 # Set BUILD64=# in the environment to disable 64-bit amd64
263 # builds on i386 machines.

265 MACH64_1=       $(MACH:sparc=sparcv9)
266 MACH64=         $(MACH64_1:i386=amd64)

268 MACH32_1=       $(MACH:sparc=sparcv7)
269 MACH32=         $(MACH32_1:i386=i86)

271 sparc_BUILD64=
272 i386_BUILD64=
273 BUILD64=        $($(MACH)_BUILD64)

275 #
276 # C compiler mode. Future compilers may change the default on us,
277 # so force extended ANSI mode globally. Lower level makefiles can
278 # override this by setting CCMODE.
279 #
280 CCMODE=                 -Xa
281 CCMODE64=               -Xa

283 #
284 # C compiler verbose mode. This is so we can enable it globally,
285 # but turn it off in the lower level makefiles of things we cannot
286 # (or aren't going to) fix.
287 #
288 CCVERBOSE=              -v

290 # set this to the secret flag "-Wc,-Qiselect-v9abiwarn=1" to get warnings
291 # from the compiler about places the -xarch=v9 may differ from -xarch=v9c.
292 V9ABIWARN=

294 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
295 # symbols (used to detect conflicts between objects that use global registers)
296 # we disable this now for safety, and because genunix doesn't link with
297 # this feature (the v9 default) enabled.
298 #
299 # REGSYM is separate since the C++ driver syntax is different.
300 CCREGSYM=               -Wc,-Qiselect-regsym=0
301 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

303 # Prevent the removal of static symbols by the SPARC code generator (cg).
304 # The x86 code generator (ube) does not remove such symbols and as such
305 # using this workaround is not applicable for x86.
306 #
307 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
308 #
309 # generate 32-bit addresses in the v9 kernel. Saves memory.
310 CCABS32=                -Wc,-xcode=abs32
311 #
312 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
313 # system calls.
314 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

316 # GCC, especially, is increasingly beginning to auto-inline functions and
317 # sadly does so separately not under the general -fno-inline-functions
318 # Additionally, we wish to prevent optimisations which cause GCC to clone
319 # functions -- in particular, these may cause unhelpful symbols to be
320 # emitted instead of function names
321 CCNOAUTOINLINE= -_gcc=-fno-inline-small-functions \
322         -_gcc=-fno-inline-functions-called-once \
323         -_gcc=-fno-ipa-cp
```



```
325 # One optimization the compiler might perform is to turn this:
326 #       #pragma weak foo
327 #       extern int foo;
328 #       if (&foo)
329 #               foo = 5;
330 # into
331 #       foo = 5;
332 # Since we do some of this (foo might be referenced in common kernel code
333 # but provided only for some cpu modules or platforms), we disable this
334 # optimization.
335 #
336 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
337 i386_CCUNBOUND  =
338 CCUNBOUND       = $($(MACH)_CCUNBOUND)

340 #
341 # compiler '-xarch' flag. This is here to centralize it and make it
342 # overridable for testing.
343 sparc_XARCH=    -m32
344 sparcv9_XARCH=  -m64
345 i386_XARCH=
346 amd64_XARCH=    -m64 -Ui386 -U__i386

348 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
349 sparc_AS_XARCH=         -xarch=v8plus
350 sparcv9_AS_XARCH=       -xarch=v9
351 i386_AS_XARCH=
352 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

354 #
355 # These flags define what we need to be 'standalone' i.e. -not- part
356 # of the rather more cosy userland environment.  This basically means
357 # the kernel.
358 #
359 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
360 #
361 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
362 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
363 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
364 # additions to SSE (SSE2, AVX ,etc.)
365 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
366 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
367 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

369 SAVEARGS=               -Wu,-save_args
370 amd64_STAND_FLAGS       += $(SAVEARGS)

372 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
373 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

375 #
376 # disable the incremental linker
377 ILDOFF=                 -xildoff
378 #
379 XDEPEND=                -xdepend
380 XFFLAG=                 -xF=%all
381 XESS=                   -xs
382 XSTRCONST=              -xstrconst

384 #
385 # turn warnings into errors (C)
386 CERRWARN = -errtags=yes -errwarn=%all
387 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
388 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

390 CERRWARN += -_gcc=-Wno-missing-braces
```

```
391 CERRWARN += -_gcc=-Wno-sign-compare
392 CERRWARN += -_gcc=-Wno-unknown-pragmas
393 CERRWARN += -_gcc=-Wno-unused-parameter
394 CERRWARN += -_gcc=-Wno-missing-field-initializers

396 # Unfortunately, this option can misfire very easily and unfixably.
397 CERRWARN +=     -_gcc=-Wno-array-bounds

399 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
400 # -nd builds
401 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
402 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

404 #
405 # turn warnings into errors (C++)
406 CCERRWARN=              -xwe

408 # C99 mode
409 C99_ENABLE=     -xc99=%all
410 C99_DISABLE=    -xc99=%none
411 C99MODE=        $(C99_DISABLE)
412 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

414 # In most places, assignments to these macros should be appended with +=
415 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
416 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
417 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
418                 $(CCSTATICSYM)
419 i386_CFLAGS=    $(i386_XARCH)
420 amd64_CFLAGS=   $(amd64_XARCH)

422 sparc_ASFLAGS=  $(sparc_AS_XARCH)
423 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
424 i386_ASFLAGS=   $(i386_AS_XARCH)
425 amd64_ASFLAGS=  $(amd64_AS_XARCH)

427 #
428 sparc_COPTFLAG=         -xO3
429 sparcv9_COPTFLAG=       -xO3
430 i386_COPTFLAG=          -O
431 amd64_COPTFLAG=         -xO3

433 COPTFLAG= $($(MACH)_COPTFLAG)
434 COPTFLAG64= $($(MACH64)_COPTFLAG)

436 # When -g is used, the compiler globalizes static objects
437 # (gives them a unique prefix). Disable that.
438 CNOGLOBAL= -W0,-noglobal

440 # Direct the Sun Studio compiler to use a static globalization prefix based on t
441 # name of the module rather than something unique. Otherwise, objects
442 # will not build deterministically, as subsequent compilations of identical
443 # source will yeild objects that always look different.
444 #
445 # In the same spirit, this will also remove the date from the N_OPT stab.
446 CGLOBALSTATIC= -W0,-xglobalstatic

448 # Sometimes we want all symbols and types in debugging information even
449 # if they aren't used.
450 CALLSYMS=       -W0,-xdbggen=no%usedonly

452 #
453 # Default debug format for Sun Studio 11 is dwarf, so force it to
454 # generate stabs.
455 #
456 DEBUGFORMAT=    -xdebugformat=stabs
```



```
458 #
459 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
460 # compilers currently prevent us from building with cc-emitted DWARF.
461 #
462 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
463 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

465 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
466 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

468 # Sun Studio produces broken userland code when saving arguments.
469 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

471 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
472 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
473 CTF_FLAGS       = $(CTF_FLAGS_32)

475 #
476 # Flags used with genoffsets
477 #
478 GOFLAGS = -_noecho \
479         $(CALLSYMS) \
480         $(CDWARFSTR)

482 OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
483         $(CC) $(GOFLAGS) $(CFLAGS) $(CPPFLAGS)

485 OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
486         $(CC) $(GOFLAGS) $(CFLAGS64) $(CPPFLAGS)

488 #
489 # tradeoff time for space (smaller is better)
490 #
491 sparc_SPACEFLAG         = -xspace -W0,-Lt
492 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
493 i386_SPACEFLAG          = -xspace
494 amd64_SPACEFLAG         =

496 SPACEFLAG               = $($(MACH)_SPACEFLAG)
497 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

499 #
500 # The Sun Studio 11 compiler has changed the behaviour of integer
501 # wrap arounds and so a flag is needed to use the legacy behaviour
502 # (without this flag panics/hangs could be exposed within the source).
503 #
504 sparc_IROPTFLAG         = -W2,-xwrap_int
505 sparcv9_IROPTFLAG       = -W2,-xwrap_int
506 i386_IROPTFLAG          =
507 amd64_IROPTFLAG         =

509 IROPTFLAG               = $($(MACH)_IROPTFLAG)
510 IROPTFLAG64             = $($(MACH64)_IROPTFLAG)

512 sparc_XREGSFLAG         = -xregs=no%appl
513 sparcv9_XREGSFLAG       = -xregs=no%appl
514 i386_XREGSFLAG          =
515 amd64_XREGSFLAG         =

517 XREGSFLAG               = $($(MACH)_XREGSFLAG)
518 XREGSFLAG64             = $($(MACH64)_XREGSFLAG)

520 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
521 # avoids stripping it.
522 SOURCEDEBUG     = $(POUND_SIGN)
```

```
523 SRCDBGBLD       = $(SOURCEDEBUG:yes=)

525 #
526 # These variables are intended ONLY for use by developers to safely pass extra
527 # flags to the compilers without unintentionally overriding Makefile-set
528 # flags.  They should NEVER be set to any value in a Makefile.
529 #
530 # They come last in the associated FLAGS variable such that they can
531 # explicitly override things if necessary, there are gaps in this, but it's
532 # the best we can manage.
533 #
534 CUSERFLAGS              =
535 CUSERFLAGS64            = $(CUSERFLAGS)
536 CCUSERFLAGS             =
537 CCUSERFLAGS64           = $(CCUSERFLAGS)

539 CSOURCEDEBUGFLAGS       =
540 CCSOURCEDEBUGFLAGS      =
541 $(SRCDBGBLD)CSOURCEDEBUGFLAGS   = -g -xs
542 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = -g -xs

544 CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
545                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
546                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
547                 $(CUSERFLAGS)
548 CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
549                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
550                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
551                 $(CUSERFLAGS64)
552 #
553 # Flags that are used to build parts of the code that are subsequently
554 # run on the build machine (also known as the NATIVE_BUILD).
555 #
556 NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
557                 $(ILDOFF) $(CERRWARN) $(C99MODE) $($(NATIVE_MACH)_CCUNBOUND) \
558                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
559                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

561 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
562 DTS_ERRNO=-D_TS_ERRNO
563 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
564         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
565         $(ADJUNCT_PROTO:%=-I%/usr/include)
566 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
567                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
568 CPPFLAGS=       $(CPPFLAGS.master)
569 AS_CPPFLAGS=    $(CPPFLAGS.master)
570 JAVAFLAGS=      -deprecation

572 #
573 # For source message catalogue
574 #
575 .SUFFIXES: $(SUFFIXES) .i .po
576 MSGROOT= $(ROOT)/catalog
577 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
578 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
579 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
580 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

582 CLOBBERFILES += $(POFILE) $(POFILES)
583 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
584 XGETTEXT= /usr/bin/xgettext
585 XGETFLAGS= -c TRANSLATION_NOTE
586 GNUXGETTEXT= /usr/gnu/bin/xgettext
587 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
588         --strict --no-location --omit-header
```



```
589 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
590         $(RM)   $@ ;\
591         $(SED) "/^domain/d" < $(<F).po > $@ ;\
592         $(RM) $(<F).po $<.i

594 #
595 # This is overwritten by local Makefile when PROG is a list.
596 #
597 POFILE= $(PROG).po

599 sparc_CCFLAGS=          -cg92 -compat=4 \
600                         -Qoption ccfe -messages=no%anachronism \
601                         $(CCERRWARN)
602 sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
603                         -Qoption ccfe -messages=no%anachronism \
604                         -Qoption ccfe -features=no%conststrings \
605                         $(CCCREGSYM) \
606                         $(CCERRWARN)
607 i386_CCFLAGS=           -compat=4 \
608                         -Qoption ccfe -messages=no%anachronism \
609                         -Qoption ccfe -features=no%conststrings \
610                         $(CCERRWARN)
611 amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
612                         -Qoption ccfe -messages=no%anachronism \
613                         -Qoption ccfe -features=no%conststrings \
614                         $(CCERRWARN)

616 sparc_CCOPTFLAG=        -O
617 sparcv9_CCOPTFLAG=      -O
618 i386_CCOPTFLAG=         -O
619 amd64_CCOPTFLAG=        -O

621 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
622 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
623 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
624                 $(CCUSERFLAGS)
625 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
626                 $(CCUSERFLAGS64)

628 #
629 #
630 #
631 ELFWRAP_FLAGS   =
632 ELFWRAP_FLAGS64 =       -64

634 #
635 # Various mapfiles that are used throughout the build, and delivered to
636 # /usr/lib/ld.
637 #
638 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
639 MAPFILE.NED_sparc =
640 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
641 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
642 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
643 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
644 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

646 #
647 # Generated mapfiles that are compiler specific, and used throughout the
648 # build.  These mapfiles are not delivered in /usr/lib/ld.
649 #
650 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
651 $(__GNUC64)MAPFILE.NGB_sparc= \
652                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
653 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
654 $(__GNUC64)MAPFILE.NGB_sparcv9= \
```

```
655                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
656 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
657 $(__GNUC64)MAPFILE.NGB_i386= \
658                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
659 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
660 $(__GNUC64)MAPFILE.NGB_amd64= \
661                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
662 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

664 #
665 # A generic interface mapfile name, used by various dynamic objects to define
666 # the interfaces and interposers the object must export.
667 #
668 MAPFILE.INT =           mapfile-intf

670 #
671 # LDLIBS32 can be set in the environment to override the following assignment.
672 # LDLIBS64 can be set to override the assignment made in Makefile.master.64.
673 # These environment settings make sure that no libraries are searched outside
674 # of the local workspace proto area:
675 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
676 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
677 #
678 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
679 LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
680 LDLIBS.cmd =    $(LDLIBS32)
681 LDLIBS.lib =    $(LDLIBS32)
682 #
683 # Define compilation macros.
684 #
685 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
686 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
687 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
688 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
689 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
690 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
691 COMPILE.d=      $(DTRACE) -G -32
692 COMPILE64.d=    $(DTRACE) -G -64
693 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
694 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

696 CLASSPATH=      .
697 COMPILE.java=   $(JAVAC) $(JAVAFLAGS) -classpath $(CLASSPATH)

699 #
700 # Link time macros
701 #
702 CCNEEDED                = -lC
703 CCEXTNEEDED             = -lCrun -lCstd
704 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -lstdc++ -lgcc_s
705 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

707 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
708 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
709 NORUNPATH=      -norunpath -nolib
710 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
711                 $(LDFLAGS) $(CCNEEDED)
712 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
713                 $(LDFLAGS) $(CCNEEDED)

715 #
716 # lint macros
717 #
718 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
719 # ON is built with a version of lint that has the fix for 4484186.
720 #
```



```
721 ALWAYS_LINT_DEFS =      -errtags=yes -s
722 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
723 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
724 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
725 ALWAYS_LINT_DEFS +=     $(C99LMODE)
726 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
727 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
728 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
729 # XX64 -- really only needed for amd64 lint
730 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
731 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
732 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
733 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
734 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
735 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
736 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
737 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

739 # This forces lint to pick up note.h and sys/note.h from Devpro rather than
740 # from the proto area.  The note.h that ON delivers would disable NOTE().
741 ONLY_LINT_DEFS =        -I$(SPRO_VROOT)/prod/include/lint

743 SECLEVEL=       core
744 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
745                 $(ALWAYS_LINT_DEFS)
746 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
747                 $(ALWAYS_LINT_DEFS)
748 LINT.s=         $(LINT.c)

750 # For some future builds, NATIVE_MACH and MACH might be different.
751 # Therefore, NATIVE_MACH needs to be redefined in the
752 # environment as ‘uname -p‘ to override this macro.
753 #
754 # For now at least, we cross-compile amd64 on i386 machines.
755 NATIVE_MACH=    $(MACH:amd64=i386)

757 # Define native compilation macros
758 #

760 # Base directory where compilers are loaded.
761 # Defined here so it can be overridden by developer.
762 #
763 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
764 SPRO_VROOT=             $(SPRO_ROOT)/SS12
765 GNU_ROOT=               $(SFW_ROOT)

767 # Till SS12u1 formally becomes the NV CBE, LINT is hard
768 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
769 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
770 # i386_LINT, amd64_LINT.
771 # Reset them when SS12u1 is rolled out.
772 #

774 # Specify platform compiler versions for languages
775 # that we use (currently only c and c++).
776 #
777 sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
778 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
779 sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
780 $(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
781 sparc_CPP=              /usr/ccs/lib/cpp
782 sparc_AS=               /usr/ccs/bin/as -xregsym=no
783 sparc_LD=               /usr/ccs/bin/ld
784 sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

786 sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
```

```
787 $(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
788 sparcv9_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
789 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
790 sparcv9_CPP=            /usr/ccs/lib/cpp
791 sparcv9_AS=             /usr/ccs/bin/as -xregsym=no
792 sparcv9_LD=             /usr/ccs/bin/ld
793 sparcv9_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

795 i386_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
796 $(__GNUC)i386_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
797 i386_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
798 $(__GNUC)i386_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
799 i386_CPP=              /usr/ccs/lib/cpp
800 i386_AS=               /usr/ccs/bin/as
801 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
802 i386_LD=               /usr/ccs/bin/ld
803 i386_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

805 amd64_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
806 $(__GNUC64)amd64_CC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
807 amd64_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
808 $(__GNUC64)amd64_CCC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
809 amd64_CPP=             /usr/ccs/lib/cpp
810 amd64_AS=              $(ONBLD_TOOLS)/bin/$(MACH)/aw
811 amd64_LD=              /usr/ccs/bin/ld
812 amd64_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

814 NATIVECC=              $($(NATIVE_MACH)_CC)
815 NATIVECCC=             $($(NATIVE_MACH)_CCC)
816 NATIVECPP=             $($(NATIVE_MACH)_CPP)
817 NATIVEAS=              $($(NATIVE_MACH)_AS)
818 NATIVELD=              $($(NATIVE_MACH)_LD)
819 NATIVELINT=            $($(NATIVE_MACH)_LINT)

821 #
822 # Makefile.master.64 overrides these settings
823 #
824 CC=                    $(NATIVECC)
825 CCC=                   $(NATIVECCC)
826 CPP=                   $(NATIVECPP)
827 AS=                    $(NATIVEAS)
828 LD=                    $(NATIVELD)
829 LINT=                  $(NATIVELINT)

831 # The real compilers used for this build
832 CW_CC_CMD=             $(CC) -_compiler
833 CW_CCC_CMD=            $(CCC) -_compiler
834 REAL_CC=               $(CW_CC_CMD:sh)
835 REAL_CCC=              $(CW_CCC_CMD:sh)

837 # Pass -Y flag to cpp (method of which is release-dependent)
838 CCYFLAG=                -Y I,

840 BDIRECT=        -Bdirect
841 BDYNAMIC=       -Bdynamic
842 BLOCAL=         -Blocal
843 BNODIRECT=      -Bnodirect
844 BREDUCE=        -Breduce
845 BSTATIC=        -Bstatic

847 ZDEFS=          -zdefs
848 ZDIRECT=        -zdirect
849 ZIGNORE=        -zignore
850 ZINITFIRST=     -zinitfirst
851 ZINTERPOSE=     -zinterpose
852 ZLAZYLOAD=      -zlazyload
```



```
853 ZLOADFLTR=      -zloadfltr
854 ZMULDEFS=       -zmuldefs
855 ZNODEFAULTLIB=  -znodefaultlib
856 ZNODEFS=        -znodefs
857 ZNODELETE=      -znodelete
858 ZNODLOPEN=      -znodlopen
859 ZNODUMP=        -znodump
860 ZNOLAZYLOAD=    -znolazyload
861 ZNOLDYNSYM=     -znoldynsym
862 ZNORELOC=       -znoreloc
863 ZNOVERSION=     -znoversion
864 ZRECORD=        -zrecord
865 ZREDLOCSYM=     -zredlocsym
866 ZTEXT=          -ztext
867 ZVERBOSE=       -zverbose

869 GSHARED=        -G
870 CCMT=           -mt

872 # Handle different PIC models on different ISAs
873 # (May be overridden by lower-level Makefiles)

875 sparc_C_PICFLAGS =      -K pic
876 sparcv9_C_PICFLAGS =    -K pic
877 i386_C_PICFLAGS =       -K pic
878 amd64_C_PICFLAGS =      -K pic
879 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
880 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

882 sparc_C_BIGPICFLAGS =   -K PIC
883 sparcv9_C_BIGPICFLAGS = -K PIC
884 i386_C_BIGPICFLAGS =    -K PIC
885 amd64_C_BIGPICFLAGS =   -K PIC
886 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
887 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

889 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
890 sparc_CC_PICFLAGS =     -Kpic
891 sparcv9_CC_PICFLAGS =   -KPIC
892 i386_CC_PICFLAGS =      -Kpic
893 amd64_CC_PICFLAGS =     -Kpic
894 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
895 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

897 AS_PICFLAGS=            $(C_PICFLAGS)
898 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

900 #
901 # Default label for CTF sections
902 #
903 CTFCVTFLAGS=            -i -L VERSION
904 $(SRCDBGBLD)CTFCVTFLAGS        += -g

906 #
907 # Override to pass module-specific flags to ctfmerge.  Currently used only by
908 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
909 # stripping.
910 #
911 CTFMRGFLAGS=
912 $(SRCDBGBLD)CTFMRGFLAGS        += -g


915 CTFCONVERT_O            = $(CTFCONVERT) $(CTFCVTFLAGS) $@

917 ELFSIGN_O=      $(TRUE)
918 ELFSIGN_CRYPTO= $(ELFSIGN_O)
```

```
ELFSIGN_OBJECT= $(ELFSIGN_O)

# Rules (normally from make.rules) and macros which are used for post
# processing files. Normally, these do stripping of the comment section
# automatically.
#    RELEASE_CM:        Should be editted to reflect the release.
#    POST_PROCESS_O:    Post-processing for '.o' files.
#    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
#    POST_PROCESS_SO:   Post-processing for '.so' files.
#    POST_PROCESS:      Post-processing for executable files (no suffix).
# Note that these macros are not completely generalized as they are to be
# used with the file name to be processed following.
#
# It is left as an exercise to Release Engineering to embellish the generation
# of the release comment string.
#
#       If this is a standard development build:
#               compress the comment section (mcs -c)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an installation build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an release build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#
# The following list of macros are used in the definition of RELEASE_CM
# which is used to label all binaries in the build:
#
#       RELEASE         Specific release of the build, eg: 5.2
#       RELEASE_MAJOR   Major version number part of $(RELEASE)
#       RELEASE_MINOR   Minor version number part of $(RELEASE)
#       VERSION         Version of the build (alpha, beta, Generic)
#       PATCHID         If this is a patch this value should contain
#                       the patchid value (eg: "Generic 100832-01"), otherwise
#                       it will be set to $(VERSION)
#       RELEASE_DATE    Date of the Release Build
#       PATCH_DATE      Date the patch was created, if this is blank it
#                       will default to the RELEASE_DATE
#
RELEASE_MAJOR=  5
RELEASE_MINOR=  11
RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
VERSION=        SunOS Development
PATCHID=        $(VERSION)
RELEASE_DATE=   release date not set
PATCH_DATE=     $(RELEASE_DATE)
RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

PROCESS_COMMENT=                    @?${MCS} -c -a $(RELEASE_CM) -a $(DEV_CM)
$(STRIP_COMMENTS)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
$(RELEASE_BUILD)PROCESS_COMMENT=    @?${MCS} -d -a $(RELEASE_CM)

STRIP_STABS=                        :
$(RELEASE_BUILD)STRIP_STABS=        $(STRIP) -x $@
$(SRCDBGBLD)STRIP_STABS=            :

POST_PROCESS_O=         $(PROCESS_COMMENT) $@
POST_PROCESS_A=
POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
                        $(ELFSIGN_OBJECT)
```



```
POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
                        $(ELFSIGN_OBJECT)

#
# chk4ubin is a tool that inspects a module for a symbol table
# ELF section size which can trigger an OBP bug on older platforms.
# This problem affects only specific sun4u bootable modules.
#
CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
CHK4UBINFLAGS=
CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

#
# PKGARCHIVE specifies the default location where packages should be
# placed if built.
#
$(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

#
# The repositories will be created with these publisher settings.  To
# update an image to the resulting repositories, this must match the
# publisher name provided to "pkg set-publisher."
#
PKGPUBLISHER_REDIST=    on-nightly
PKGPUBLISHER_NONREDIST= on-extra

#       Default build rules which perform comment section post-processing.
#
.c:
        $(LINK.c) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.c.o:
        $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
.c.a:
        $(COMPILE.c) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.s.o:
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)
.s.a:
        $(COMPILE.s) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.cc:
        $(LINK.cc) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.cc.o:
        $(COMPILE.cc) $(OUTPUT_OPTION) $<
        $(POST_PROCESS_O)
.cc.a:
        $(COMPILE.cc) -o $% $<
        $(AR) $(ARFLAGS) $@ $%
        $(PROCESS_COMMENT) $%
        $(RM) $%
.y:
        $(YACC.y) $<
        $(LINK.c) -o $@ y.tab.c $(LDLIBS)
        $(POST_PROCESS)
        $(RM) y.tab.c
.y.o:
        $(YACC.y) $<
```

```
1051         $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1052         $(POST_PROCESS_O)
1053         $(RM) y.tab.c
1054 .l:
1055         $(RM) $*.c
1056         $(LEX.l) $< > $*.c
1057         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1058         $(POST_PROCESS)
1059         $(RM) $*.c
1060 .l.o:
1061         $(RM) $*.c
1062         $(LEX.l) $< > $*.c
1063         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1064         $(POST_PROCESS_O)
1065         $(RM) $*.c

1067 .bin.o:
1068         $(COMPILE.b) -o $@ $<
1069         $(POST_PROCESS_O)

1071 .java.class:
1072         $(COMPILE.java) $<

1074 # Bourne and Korn shell script message catalog build rules.
1075 # We extract all gettext strings with sed(1) (being careful to permit
1076 # multiple gettext strings on the same line), weed out the dups, and
1077 # build the catalogue with awk(1).

1079 .sh.po .ksh.po:
1080         $(SED) -n -e ":a"                                   \
1081                   -e "h"                                    \
1082                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"   \
1083                   -e "x"                                    \
1084                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1085                   -e "t a"                                  \
1086                $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1088 #
1089 # Python and Perl executable and message catalog build rules.
1090 #
1091 .SUFFIXES: .pl .pm .py .pyc

1093 .pl:
1094         $(RM) $@;
1095         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1096         $(CHMOD) +x $@

1098 .py:
1099         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1101 .py.pyc:
1102         $(RM) $@
1103         $(PYTHON) -mpy_compile $<
1104         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1106 .py.po:
1107         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1109 .pl.po .pm.po:
1110         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1111         $(RM)   $@ ;
1112         $(SED) "/^domain/d" < $(<F).po > $@ ;
1113         $(RM) $(<F).po

1115 #
1116 # When using xgettext, we want messages to go to the default domain,
```



```
1117 # rather than the specified one.  This special version of the
1118 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1119 # causing xgettext to put all messages into the default domain.
1120 #
1121 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1123 .c.i:
1124         $(CPPFORPO) $< > $@

1126 .h.i:
1127         $(CPPFORPO) $< > $@

1129 .y.i:
1130         $(YACC) -d $<
1131         $(CPPFORPO) y.tab.c  > $@
1132         $(RM) y.tab.c

1134 .l.i:
1135         $(LEX) $<
1136         $(CPPFORPO) lex.yy.c  > $@
1137         $(RM) lex.yy.c

1139 .c.po:
1140         $(CPPFORPO) $< > $<.i
1141         $(BUILD.po)

1143 .y.po:
1144         $(YACC) -d $<
1145         $(CPPFORPO) y.tab.c  > $<.i
1146         $(BUILD.po)
1147         $(RM) y.tab.c

1149 .l.po:
1150         $(LEX) $<
1151         $(CPPFORPO) lex.yy.c  > $<.i
1152         $(BUILD.po)
1153         $(RM) lex.yy.c

1155 #
1156 # Rules to perform stylistic checks
1157 #
1158 .SUFFIXES: .x .xml .check .xmlchk

1160 .h.check:
1161         $(DOT_H_CHECK)

1163 .x.check:
1164         $(DOT_X_CHECK)

1166 .xml.xmlchk:
1167         $(MANIFEST_CHECK)

1169 #
1170 # Include rules to render automated sccs get rules "safe".
1171 #
1172 include $(SRC)/Makefile.noget
```

```
**********************************************************
   10884 Thu Dec 26 13:48:09 2013
new/usr/src/cmd/Makefile
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
# Copyright (c) 2012 Joyent, Inc.  All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright (c) 2013 DEY Storage Systems, Inc. All rights reserved.

include ../Makefile.master

#
#       Note that the commands 'agents', 'lp', 'perl', and 'man' are first in
#       the list, violating alphabetical order.  This is because they are very
#       long-running and should be given the most wall-clock time for a
#       parallel build.
#
#       Commands in the FIRST_SUBDIRS list are built before starting the build
#       of other commands.  Currently this includes only 'isaexec' and
#       'platexec'.  This is necessary because $(ROOT)/usr/lib/isaexec or
#       $(ROOT)/usr/lib/platexec must exist when some other commands are built
#       because their 'make install' creates a hard link to one of them.
#
#       Commands are listed one per line so that TeamWare can auto-merge most
#       changes.
#

FIRST_SUBDIRS=          \
        isaexec         \
        platexec

COMMON_SUBDIRS=         \
        allocate        \
        availdevs       \
        lp              \
        perl            \
        man             \
        Adm             \
        abi             \
        adbgen          \
        acct            \
        acctadm         \
        acpitools       \
```



```
        arch            \
        asa             \
        ast             \
        audio           \
        auths           \
        autopush        \
        avs             \
        awk             \
        awk_xpg4        \
        backup          \
        banner          \
        bart            \
        basename        \
        bc              \
        bdiff           \
        beadm           \
        bfs             \
        bnu             \
        boot            \
        busstat         \
        cal             \
        calendar        \
        captoinfo       \
        cat             \
        cdrw            \
        cfgadm          \
        checkeq         \
        checknr         \
        chgrp           \
        chmod           \
        chown           \
        chroot          \
        clear           \
        clinfo          \
        cmd-crypto      \
        cmd-inet        \
        col             \
        compress        \
        consadm         \
        coreadm         \
        cpio            \
        cpc             \
        cron            \
        crypt           \
        csh             \
        csplit          \
        ctrun           \
        ctstat          \
        ctwatch         \
        datadm          \
        date            \
        dc              \
        dd              \
        deroff          \
        devfsadm        \
        syseventd       \
        devctl          \
        devinfo         \
        devmgmt         \
        devprop         \
        dfs.cmds        \
        diff            \
        diff3           \
        diffmk          \
        dircmp          \
        dirname         \
```

```
128         dis             \
129         diskmgtd        \
130         dispadmin       \
131         dladm           \
132         dlstat          \
133         dmesg           \
134         dodatadm        \
135         dtrace          \
136         du              \
137         dumpadm         \
138         dumpcs          \
139         echo            \
140         ed              \
141         eeprom          \
142         egrep           \
143         eject           \
144         emul64ioctl     \
145         enhance         \
146         env             \
147         eqn             \
148         expand          \
149         expr            \
150         exstr           \
151         factor          \
152         false           \
153         fcinfo          \
154         fcoesvc         \
155         fdetach         \
156         fdformat        \
157         fdisk           \
158         filesync        \
159         fgrep           \
160         file            \
161         filebench       \
162         find            \
163         flowadm         \
164         flowstat        \
165         fm              \
166         fmt             \
167         fmthard         \
168         fmtmsg          \
169         fold            \
170         format          \
171         fs.d            \
172         fstyp           \
173         fuser           \
174         fwflash         \
175         gcore           \
176         gencat          \
177         geniconvtbl     \
178         genmsg          \
179         getconf         \
180         getdevpolicy    \
181         getent          \
182         getfacl         \
183         getmajor        \
184         getopt          \
185         gettext         \
186         gettxt          \
187         grep            \
188         grep_xpg4       \
189         groups          \
190         grpck           \
191         gss             \
192         hal             \
193         halt            \
```



```
194         head            \
195         hostid          \
196         hostname        \
197         hotplug         \
198         hotplugd        \
199         hwdata          \
200         ibd_upgrade     \
201         id              \
202         idmap           \
203         infocmp         \
204         init            \
205         initpkg         \
206         install.d       \
207         intrd           \
208         intrstat        \
209         ipcrm           \
210         ipcs            \
211         ipdadm          \
212         ipf             \
213         isainfo         \
214         isalist         \
215         itutools        \
216         iscsiadm        \
217         iscsid          \
218         iscsitsvc       \
219         isns            \
220         itadm           \
221         java            \
222         kbd             \
223         keyserv         \
224         killall         \
225         krb5            \
226         ksh             \
227         kvmstat         \
228         last            \
229         lastcomm        \
230         latencytop      \
231         ldap            \
232         ldapcachemgr    \
233         lgrpinfo        \
234         line            \
235         link            \
236         dlmgmtd         \
237         listen          \
238         loadkeys        \
239         locale          \
240         localedef       \
241         lockstat        \
242         locator         \
243         lofiadm         \
244         logadm          \
245         logger          \
246         login           \
247         logins          \
248         look            \
249         ls              \
250         luxadm          \
251         lvm             \
252         mach            \
253         machid          \
254         mail            \
255         mailx           \
256         makekey         \
257         mdb             \
258         mesg            \
259         mkdir           \
```

```
260         mkfifo          \
261         mkfile          \
262         mkmsgs          \
263         mknod           \
264         mkpwdict        \
265         mktemp          \
266         modload         \
267         more            \
268         mpathadm        \
269         msgfmt          \
270         msgid           \
271         mt              \
272         mv              \
273         mvdir           \
274         ndmpadm         \
275         ndmpd           \
276         ndmpstat        \
277         netadm          \
278         netfiles        \
279         newform         \
280         newgrp          \
281         news            \
282         newtask         \
283         nice            \
284         nl              \
285         nlsadmin        \
286         nohup           \
287         nsadmin         \
288         nscd            \
289         oamuser         \
290         oawk            \
291         od              \
292         pack            \
293         pagesize        \
294         passmgmt        \
295         passwd          \
296         pathchk         \
297         pbind           \
298         pcidr           \
299         pcitool         \
300         pfexec          \
301         pfexecd         \
302         pginfo          \
303         pgstat          \
304         pgrep           \
305         picl            \
306         plimit          \
307         policykit       \
308         pools           \
309         power           \
310         powertop        \
311         ppgsz           \
312         pg              \
313         plockstat       \
314         pr              \
315         prctl           \
316         print           \
317         printf          \
318         priocntl        \
319         profiles        \
320         projadd         \
321         projects        \
322         prstat          \
323         prtconf         \
324         prtdiag         \
325         prtvtoc         \
```



```
326         ps              \
327         psradm          \
328         psrinfo         \
329         psrset          \
330         ptools          \
331         pwck            \
332         pwconv          \
333         pwd             \
334         pyzfs           \
335         raidctl         \
336         ramdiskadm      \
337         rcap            \
338         rcm_daemon      \
339         rctladm         \
340         refer           \
341         regcmp          \
342         renice          \
343         rexd            \
344         rm              \
345         rmdir           \
346         rmformat        \
347         rmmount         \
348         rmt             \
349         rmvolmgr        \
350         roles           \
351         rpcbind         \
352         rpcgen          \
353         rpcinfo         \
354         rpcsvc          \
355         runat           \
356         sa              \
357         saf             \
358         sasinfo         \
359         savecore        \
360         sbdadm          \
361         script          \
362         scsi            \
363         sdiff           \
364         sdpadm          \
365         sed             \
366         sendmail        \
367         setfacl         \
368         setmnt          \
369         setpgrp         \
370         setuname        \
371         sgs             \
372         sh              \
373         shcomp          \
374         smbios          \
375         smbsrv          \
376         smserverd       \
377         soelim          \
378         sort            \
379         spell           \
380         split           \
381         sqlite          \
382         srchtxt         \
383         srptadm         \
384         srptsvc         \
385         ssh             \
386         stat            \
387         stmfadm         \
388         stmfproxy       \
389         stmfsvc         \
390         stmsboot        \
391         streams         \
```

```
392         strings         \
393         su              \
394         sulogin         \
395         sunpc           \
396         svc             \
397         svr4pkg         \
398         swap            \
399         sync            \
400         sysdef          \
401         syseventadm     \
402         syslogd         \
403         tabs            \
404         tail            \
405         tar             \
406         tbl             \
407         tcopy           \
408         tcpd            \
409         terminfo        \
410         th_tools        \
411         tic             \
412         time            \
413         tip             \
414         tnf             \
415         touch           \
416         tput            \
417         tr              \
418         trapstat        \
419         troff           \
420         true            \
421         truss           \
422         tsol            \
423         tty             \
424         ttymon          \
425         tzreload        \
426         uadmin          \
427         ul              \
428         uname           \
429         units           \
430         unlink          \
431         unpack          \
432         userattr        \
433         users           \
434         utmp_update     \
435         utmpd           \
436         valtools        \
437         vgrind          \
438         vi              \
439         volcheck        \
440         volrmmount      \
441         vrrpadm         \
442         vscan           \
443         vt              \
444         w               \
445         wall            \
446         which           \
447         who             \
448         whodo           \
449         wracct          \
450         write           \
451         wusbadm         \
452         xargs           \
453         xstr            \
454         yes             \
455         ypcmd           \
456         yppasswd        \
457         zdb             \
```



```
458         zdump           \
459         zfs             \
460         zhack           \
461         zic             \
462         zinject         \
463         zlogin          \
464         zoneadm         \
465         zoneadmd        \
466         zonecfg         \
467         zonename        \
468         zpool           \
469         zlook           \
470         zonestat        \
471         zstreamdump     \
472         ztest

474 i386_SUBDIRS=           \
475         acpihpd         \
476         addbadsec       \
477         biosdev         \
478         diskscan        \
479         lms             \
480         ntfsprogs       \
481         parted          \
482         rtc             \
483         ucodeadm        \
484         xvm

486 sparc_SUBDIRS=          \
487         cvcd            \
488         dcs             \
489         device_remap    \
490         drd             \
491         fruadm          \
492         ldmad           \
493         oplhpd          \
494         prtdscp         \
495         prtfru          \
496         scadm           \
497         sckmd           \
498         sf880drd        \
499         virtinfo        \
500         vntsd

502 #
503 # Commands that are messaged.  Note that 'lp' and 'man' come first
504 # (see previous comment about 'lp' and 'man').
505 #
506 MSGSUBDIRS=             \
507         lp              \
508         man             \
509         abi             \
510         acctadm         \
511         allocate        \
512         asa             \
513         audio           \
514         audit           \
515         auditconfig     \
516         auditd          \
517         auditrecord     \
518         auditset        \
519         auths           \
520         autopush        \
521         avs             \
522         awk             \
523         awk_xpg4        \
```

```
524         backup          \
525         banner          \
526         bart            \
527         basename        \
528         beadm           \
529         bnu             \
530         busstat         \
531         cal             \
532         cat             \
533         cdrw            \
534         cfgadm          \
535         checkeq         \
536         checknr         \
537         chgrp           \
538         chmod           \
539         chown           \
540         cmd-crypto      \
541         cmd-inet        \
542         col             \
543         compress        \
544         consadm         \
545         coreadm         \
546         cpio            \
547         cpc             \
548         cron            \
549         csh             \
550         csplit          \
551         ctrun           \
552         ctstat          \
553         ctwatch         \
554         datadm          \
555         date            \
556         dc              \
557         dcs             \
558         dd              \
559         deroff          \
560         devfsadm        \
561         dfs.cmds        \
562         diff            \
563         diffmk          \
564         dladm           \
565         dlstat          \
566         du              \
567         dumpcs          \
568         ed              \
569         eject           \
570         env             \
571         eqn             \
572         expand          \
573         expr            \
574         fcinfo          \
575         fgrep           \
576         file            \
577         filesync        \
578         find            \
579         flowadm         \
580         flowstat        \
581         fm              \
582         fold            \
583         fs.d            \
584         fwflash         \
585         geniconvtbl     \
586         genmsg          \
587         getconf         \
588         getent          \
589         gettext         \
```



```
590         gettxt          \
591         grep            \
592         grep_xpg4       \
593         grpck           \
594         gss             \
595         halt            \
596         head            \
597         hostname        \
598         hotplug         \
599         id              \
600         idmap           \
601         isaexec         \
602         iscsiadm        \
603         iscsid          \
604         isns            \
605         itadm           \
606         kbd             \
607         krb5            \
608         ksh             \
609         last            \
610         ldap            \
611         ldapcachemgr    \
612         lgrpinfo        \
613         locale          \
614         lofiadm         \
615         logadm          \
616         logger          \
617         logins          \
618         ls              \
619         luxadm          \
620         lvm             \
621         mailx           \
622         mesg            \
623         mkdir           \
624         mkpwdict        \
625         mktemp          \
626         more            \
627         mpathadm        \
628         msgfmt          \
629         mv              \
630         ndmpadm         \
631         ndmpstat        \
632         newgrp          \
633         newtask         \
634         nice            \
635         nohup           \
636         oawk            \
637         pack            \
638         passwd          \
639         passmgmt        \
640         pathchk         \
641         pfexec          \
642         pg              \
643         pgrep           \
644         picl            \
645         pools           \
646         power           \
647         pr              \
648         praudit         \
649         print           \
650         profiles        \
651         projadd         \
652         projects        \
653         prstat          \
654         prtdiag         \
655         ps              \
```

```
656         psrinfo         \
657         ptools          \
658         pwconv          \
659         pwd             \
660         pyzfs           \
661         raidctl         \
662         ramdiskadm      \
663         rcap            \
664         rcm_daemon      \
665         refer           \
666         regcmp          \
667         renice          \
668         roles           \
669         rm              \
670         rmdir           \
671         rmformat        \
672         rmmount         \
673         rmvolmgr        \
674         sasinfo         \
675         sbdadm          \
676         scadm           \
677         script          \
678         scsi            \
679         sdiff           \
680         sdpadm          \
681         sgs             \
682         sh              \
683         shcomp          \
684         smbsrv          \
685         sort            \
686         split           \
687         srptadm         \
688         ssh             \
689         stat            \
690         stmfadm         \
691         stmsboot        \
692         strings         \
693         su              \
694         svc             \
695         svr4pkg         \
696         swap            \
697         syseventadm     \
698         syseventd       \
699         tabs            \
700         tar             \
701         tbl             \
702         time            \
703         tnf             \
704         touch           \
705         tput            \
706         troff           \
707         tsol            \
708         tty             \
709         ttymon          \
710         tzreload        \
711         ul              \
712         uname           \
713         units           \
714         unlink          \
715         unpack          \
716         userattr        \
717         valtools        \
718         vgrind          \
719         vi              \
720         volcheck        \
721         volrmmount      \
```



```
722         vrrpadm         \
723         vscan           \
724         w               \
725         who             \
726         whodo           \
727         wracct          \
728         write           \
729         wusbadm         \
730         xargs           \
731         yppasswd        \
732         zdump           \
733         zfs             \
734         zic             \
735         zlogin          \
736         zoneadm         \
737         zoneadmd        \
738         zonecfg         \
739         zonename        \
740         zpool           \
741         zonestat

743 sparc_MSGSUBDIRS=       \
744         fruadm          \
745         prtdscp         \
746         prtfru          \
747         virtinfo        \
748         vntsd

750 i386_MSGSUBDIRS=        \
751         ucodeadm

753 #
754 # commands that use dcgettext for localized time, LC_TIME
755 #
756 DCSUBDIRS=              \
757         cal             \
758         cfgadm          \
759         diff            \
760         ls              \
761         pr              \
762         ps              \
763         tar             \
764         w               \
765         who             \
766         whodo           \
767         write

769 #
770 # commands that belong only to audit.
771 #
772 AUDITSUBDIRS=           \
773         amt             \
774         audit           \
775         audit_warn      \
776         auditconfig     \
777         auditd          \
778         auditrecord     \
779         auditreduce     \
780         auditset        \
781         auditstat       \
782         praudit

784 #
785 # commands not owned by the systems group
786 #
787 BWOSDIRS=
```

```
790 all :=          TARGET = all
791 install :=      TARGET = install
792 clean :=        TARGET = clean
793 clobber :=      TARGET = clobber
794 lint :=         TARGET = lint
795 _msg :=         TARGET = _msg
796 _dc :=          TARGET = _dc

798 .KEEP_STATE:

800 SUBDIRS = $(COMMON_SUBDIRS) $($(MACH)_SUBDIRS)

802 .PARALLEL:      $(BWOSDIRS) $(SUBDIRS) $(MSGSUBDIRS) $(AUDITSUBDIRS)

804 all install clean clobber lint: $(FIRST_SUBDIRS) .WAIT $(SUBDIRS) \
805         $(AUDITSUBDIRS)

807 #
808 # Manifests cannot be checked in parallel, because we are using
809 # the global repository that is in $(SRC)/cmd/svc/seed/global.db.
810 # For this reason, to avoid .PARALLEL and .NO_PARALLEL conflicts,
811 # we spawn off a sub-make to perform the non-parallel 'make check'
812 #
813 check:
814         $(MAKE) -f Makefile.check check

816 #
817 # The .WAIT directive works around an apparent bug in parallel make.
818 # Evidently make was getting the target _msg vs. _dc confused under
819 # some level of parallelization, causing some of the _dc objects
820 # not to be built.
821 #
822 _msg: $(MSGSUBDIRS) $($(MACH)_MSGSUBDIRS) .WAIT _dc

824 _dc: $(DCSUBDIRS)

826 #
827 # Dependencies
828 #
829 fs.d: fstyp
830 ksh:    shcomp isaexec
831 mdb:    terminfo
832 print:  lp

834 $(FIRST_SUBDIRS) $(BWOSDIRS) $(SUBDIRS) $(AUDITSUBDIRS): FRC
835         @if [ -f $@/Makefile  ]; then \
836                 cd $@; pwd; $(MAKE) $(TARGET); \
837         else \
838                 true; \
839         fi

841 FRC:
```

****
786 Thu Dec 26 13:48:09 2013
new/usr/src/cmd/acpitools/Makefile
PANKOVs restructure
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

include         $(SRC)/cmd/Makefile.cmd

SUBDIRS=        acpiexec acpixtract iasl

all             := TARGET = all
clean           := TARGET = clean
clobber         := TARGET = clobber
install         := TARGET = install

.KEEP_STATE:

all clean clobber install: $(SUBDIRS)

$(SUBDIRS):     FRC
                @cd $@; pwd; $(MAKE) $(TARGET)

include         $(SRC)/cmd/Makefile.targ

FRC:
```

```
**********************************************************
    4460 Thu Dec 26 13:48:09 2013
new/usr/src/cmd/acpitools/acpiexec/Makefile
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

PROG=           acpiexec
OBJS=           oslstubs.o
# common
OBJS +=         getopt.o acgetline.o

# components/debugger
OBJS +=         dbcmds.o dbdisply.o dbexec.o dbfileio.o dbhistry.o      \
                dbinput.o dbmethod.o dbnames.o dbstats.o dbutils.o      \
                dbxface.o dbconvert.o
# components/disassembler
OBJS +=         dmbuffer.o dmnames.o dmobject.o dmopcode.o dmresrc.o    \
                dmresrcl.o dmresrcs.o dmutils.o dmwalk.o dmdeferred.o   \
                dmresrcl2.o
# components/dispatcher
OBJS +=         dsargs.o dscontrol.o dsfield.o dsinit.o dsmethod.o      \
                dsmthdat.o dsobject.o dsopcode.o dsutils.o dswexec.o    \
                dswload.o dswload2.o dswscope.o dswstate.o
# components/events
OBJS +=         evevent.o evglock.o evgpe.o evgpeblk.o evgpeinit.o      \
                evgpeutil.o evmisc.o evregion.o evrgnini.o evsci.o      \
                evxface.o evxfevnt.o evxfgpe.o evxfregn.o evhandler.o
# components/executer
OBJS +=         exconfig.o exconvrt.o excreate.o exdebug.o exdump.o     \
                exfield.o exfldio.o exmisc.o exmutex.o exnames.o        \
                exoparg1.o exoparg2.o exoparg3.o exoparg6.o exprep.o    \
                exregion.o exresnte.o exresolv.o exresop.o exstore.o    \
                exstoren.o exstorob.o exsystem.o exutils.o
# components/hardware
OBJS +=         hwacpi.o hwgpe.o hwpci.o hwregs.o hwsleep.o hwvalid.o   \
                hwxface.o hwesleep.o hwxfsleep.o
# components/namespace
OBJS +=         nsaccess.o nsalloc.o nsdump.o nsdumpdv.o nseval.o       \
                nsinit.o nsload.o nsnames.o nsobject.o nsparse.o        \
                nspredef.o nsrepair.o nsrepair2.o nssearch.o nsutils.o  \
                nswalk.o nsxfeval.o nsxfname.o nsxfobj.o nsarguments.o  \
                nsconvert.o nsprepkg.o

# components/parser
OBJS +=         psargs.o psloop.o psopcode.o psparse.o psscope.o        \
                pstree.o psutils.o pswalk.o psxface.o psobject.o psopinfo.o
# components/resources
OBJS +=         rsaddr.o rscalc.o rscreate.o rsdump.o rsinfo.o rsio.o   \
                rsirq.o rslist.o rsmemory.o rsmisc.o rsutils.o          \
                rsxface.o rsdumpinfo.o rsserial.o
```



```
# components/tables
OBJS +=         tbfadt.o tbfind.o tbinstal.o tbutils.o tbxface.o        \
                tbxfroot.o tbprint.o tbxfload.o
# components/utilities
OBJS +=         utalloc.o utcache.o utcopy.o utdebug.o utdecode.o       \
                utdelete.o uteval.o utglobal.o utids.o utinit.o         \
                utlock.o utmath.o utmisc.o utmutex.o utobject.o         \
                utresrc.o utstate.o uttrack.o utosi.o utxferror.o       \
                utxface.o utaddress.o utbuffer.o uterror.o utexcep.o    \
                utownerid.o utpredef.o utstring.o utxfinit.o utxfmutex.o
# os_specific/service_layers
OBJS +=         osunixxf.o
# tools/acpiexec
OBJS +=         aeexec.o aehandlers.o aemain.o aetables.o

ACPICA=         $(SRC)/common/acpica

include         $(SRC)/cmd/Makefile.cmd

C99MODE=        $(C99_ENABLE)
CPPFLAGS +=     -DACPI_EXEC_APP -I$(ACPICA)/include -I$(ACPICA)/compiler

.KEEP_STATE:

all:            $(PROG)

clean:
                $(RM) $(OBJS)

install:        all $(ROOTPROG)

$(PROG):        $(OBJS)
                $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
                $(POST_PROCESS)

%.o:            $(ACPICA)/common/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/compiler/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/debugger/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/disassembler/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/dispatcher/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/events/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/executer/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/hardware/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)
```

```
127 %.o:            $(ACPICA)/components/namespace/%.c
128                 $(COMPILE.c) -o $@ $<
129                 $(POST_PROCESS_O)

131 %.o:            $(ACPICA)/components/parser/%.c
132                 $(COMPILE.c) -o $@ $<
133                 $(POST_PROCESS_O)

135 %.o:            $(ACPICA)/components/resources/%.c
136                 $(COMPILE.c) -o $@ $<
137                 $(POST_PROCESS_O)

139 %.o:            $(ACPICA)/components/tables/%.c
140                 $(COMPILE.c) -o $@ $<
141                 $(POST_PROCESS_O)

143 %.o:            $(ACPICA)/components/utilities/%.c
144                 $(COMPILE.c) -o $@ $<
145                 $(POST_PROCESS_O)

147 %.o:            $(ACPICA)/os_specific/service_layers/%.c
148                 $(COMPILE.c) -o $@ $<
149                 $(POST_PROCESS_O)

151 %.o:            $(ACPICA)/tools/acpiexec/%.c
152                 $(COMPILE.c) -o $@ $<
153                 $(POST_PROCESS_O)

155 include        $(SRC)/cmd/Makefile.targ
```

****
681 Thu Dec 26 13:48:09 2013
new/usr/src/cmd/acpitools/acpiexec/oslstubs.c
PANKOVs restructure
****

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * OSL stubs
 */

#include <acpi.h>

UINT32
__acpi_acquire_global_lock(void *Facs)
{
        return (0);
}

UINT32
__acpi_release_global_lock(void *Facs)
{
        return (0);
}

void
__acpi_wbinvd(void)
{
}
```

```
**********************************************************
    2989 Thu Dec 26 13:48:10 2013
new/usr/src/cmd/acpitools/acpinames/Makefile
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.   All rights reserved.
#

PROG=           acpinames
# common
OBJS +=         getopt.o
#
OBJS +=         anmain.o anstubs.o antables.o
OBJS +=         dbfileio.o
OBJS +=         dsfield.o dsmthdat.o dsobject.o dsutils.o dswload.o     \
                dswload2.o dswscope.o dswstate.o
OBJS +=         excreate.o exnames.o exresnte.o exresolv.o exutils.o
OBJS +=         nsaccess.o nsalloc.o nsdump.o nsinit.o nsload.o         \
                nsnames.o nsobject.o nsparse.o nssearch.o nsutils.o     \
                nswalk.o nsxfeval.o nsxfname.o nsxfobj.o
OBJS +=         osunixxf.o
OBJS +=         psargs.o psloop.o psopcode.o psparse.o psscope.o        \
                pstree.o psutils.o pswalk.o psxface.o psobject.o psopinfo.o
OBJS +=         tbfadt.o tbfind.o tbinstal.o tbutils.o tbxface.o        \
                tbxfroot.o tbprint.o tbxfload.o
OBJS +=         utalloc.o utcache.o utdebug.o utdecode.o utdelete.o     \
                utglobal.o utlock.o utmath.o utmisc.o utmutex.o         \
                utobject.o utstate.o utosi.o utxferror.o utxface.o      \
                utaddress.o uterror.o utexcep.o utownerid.o utstring.o  \
                utxfinit.o

ACPICA=         $(SRC)/common/acpica

include         $(SRC)/cmd/Makefile.cmd

C99MODE=        $(C99_ENABLE)
CPPFLAGS +=     -DACPI_NAMES_APP \
                -I$(ACPICA)/include -I$(ACPICA)/tools/acpinames

.KEEP_STATE:

all:            $(PROG)

clean:
                $(RM) $(OBJS)

install:        all $(ROOTPROG)

$(PROG):        $(OBJS)
                $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
                $(POST_PROCESS)
```



```
%.o:            $(ACPICA)/common/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/compiler/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/debugger/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/disassembler/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/dispatcher/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/events/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/executer/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/hardware/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/namespace/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/parser/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/resources/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/tables/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/components/utilities/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/os_specific/service_layers/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/tools/acpinames/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

include         $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    1004 Thu Dec 26 13:48:10 2013
new/usr/src/cmd/acpitools/acpixtract/Makefile
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

PROG=           acpixtract
OBJS=           acpixtract.o axmain.o getopt.o

ACPICA=         $(SRC)/common/acpica

include         $(SRC)/cmd/Makefile.cmd

C99MODE=        $(C99_ENABLE)
CPPFLAGS +=     -DACPI_XTRACT_APP -I$(ACPICA)/include

.KEEP_STATE:

$(PROG):        $(OBJS)
                $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
                $(POST_PROCESS)

%.o:            $(ACPICA)/common/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

%.o:            $(ACPICA)/tools/acpixtract/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)

all:            $(PROG)

clean:
                $(RM) $(OBJS)

install:        all $(ROOTPROG)

include         $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    5193 Thu Dec 26 13:48:10 2013
new/usr/src/cmd/acpitools/iasl/Makefile
fix typo in CLEANFILES for acpitools/iasl
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

PROG=           iasl
# common
OBJS +=         adfile.o adisasm.o adwalk.o dmextern.o  \
                dmrestag.o dmtable.o dmtbdump.o dmtbinfo.o getopt.o     \
                ahpredef.o
# compiler (generated)
OBJS +=         aslcompilerparse.o aslcompilerlex.o dtparserparse.o     \
                dtparserlex.o prexpress.o prmacros.o prscan.o prutils.o \
                prparserparse.o prparserlex.o
# compiler
OBJS +=         aslanalyze.o aslbtypes.o aslcodegen.o aslcompile.o      \
                aslerror.o aslfiles.o aslfold.o asllength.o             \
                asllisting.o aslload.o asllookup.o aslmain.o aslmap.o   \
                aslopcodes.o asloperands.o aslopt.o aslpredef.o         \
                aslresource.o aslrestype1.o aslrestype1i.o              \
                aslrestype2.o aslrestype2d.o aslrestype2e.o             \
                aslrestype2q.o aslrestype2w.o aslstartup.o aslstubs.o   \
                asltransform.o asltree.o aslutils.o asluuid.o           \
                aslwalks.o dtcompile.o dtexpress.o dtfield.o dtio.o     \
                dtsubtable.o dttable.o dttemplate.o dtutils.o aslfileio.o
                aslhex.o asllistsup.o aslmethod.o aslnamesp.o asloffset.o
                asloptions.o aslprepkg.o aslrestype2s.o aslxref.o
# components/debugger
OBJS +=         dbfileio.o
# components/disassembler
OBJS +=         dmbuffer.o dmnames.o dmopcode.o dmresrc.o dmresrcl.o    \
                dmresrcs.o dmutils.o dmwalk.o dmdeferred.o dmobject.o
                dmresrcl2.o

# components/dispatcher
OBJS +=         dsargs.o dscontrol.o dsfield.o dsobject.o dsopcode.o    \
                dsutils.o dswexec.o dswload.o dswload2.o dswscope.o     \
                dswstate.o
# components/executer
OBJS +=         exconvrt.o excreate.o exdump.o exmisc.o exmutex.o       \
                exnames.o exoparg1.o exoparg2.o exoparg3.o exoparg6.o   \
                exprep.o exresnte.o exresolv.o exresop.o exstore.o      \
                exstoren.o exstorob.o exsystem.o exutils.o exregion.o
# components/namespace
OBJS +=         nsaccess.o nsalloc.o nsdump.o nsnames.o nsobject.o      \
                nsparse.o nssearch.o nsutils.o nswalk.o nsxfobj.o
# components/parser
OBJS +=         psargs.o psloop.o psopcode.o psparse.o psscope.o        \
                pstree.o psutils.o pswalk.o psobject.o psopinfo.o
```



```
# components/tables
OBJS +=         tbfadt.o tbinstal.o tbutils.o tbxface.o tbprint.o
# components/utilities
OBJS +=         utalloc.o utcache.o utcopy.o utdebug.o utdecode.o       \
                utdelete.o utglobal.o utinit.o utlock.o utmath.o        \
                utmisc.o utmutex.o utobject.o utresrc.o utstate.o       \
                utxface.o utxferror.o utaddress.o utbuffer.o uterror.o  \
                utexcep.o utownerid.o utpredef.o utstring.o

# os_specific/service_layers
OBJS +=         osunixxf.o

CLEANFILES=     aslcompiler.y.h aslcompilerlex.c aslcompilerparse.c     \
                dtparser.y.h dtparserlex.c dtparserparse.c prparser.y.h \
                prparserlex.c prparserparse.c

ACPICA=         $(SRC)/common/acpica

include         $(SRC)/cmd/Makefile.cmd

C99MODE=        $(C99_ENABLE)
CPPFLAGS +=     -DACPI_ASL_COMPILER -DACPI_USE_STANDARD_HEADERS         \
                -I$(ACPICA)/include -I$(ACPICA)/compiler -I.

# Flags for flex and bison
LFLAGS=         -i -s
YFLAGS=         -d

.KEEP_STATE:

all:            $(PROG)

clean:
                $(RM) $(CLEANFILES) $(OBJS)

install:        all $(ROOTPROG)

$(PROG):        $(OBJS)
                $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
                $(POST_PROCESS)

aslcompilerlex.c: $(ACPICA)/compiler/aslcompiler.l
                $(FLEX) $(LFLAGS) -PAslCompiler -o$@ $(ACPICA)/compiler/aslcompi

aslcompilerparse.c: $(ACPICA)/compiler/aslcompiler.y
                $(BISON) $(YFLAGS) -pAslCompiler -o$@ $(ACPICA)/compiler/aslcomp
                $(RM) aslcompiler.y.h; $(MV) aslcompilerparse.h aslcompiler.y.h

dtparserlex.c:  $(ACPICA)/compiler/dtparser.l
                $(FLEX) $(LFLAGS) -PDtParser -o$@ $(ACPICA)/compiler/dtparser.l

dtparserparse.c: $(ACPICA)/compiler/dtparser.y
                $(BISON) $(YFLAGS) -pDtParser -o$@ $(ACPICA)/compiler/dtparser.y
                $(RM) dtparser.y.h; $(MV) dtparserparse.h dtparser.y.h

prparserlex.c:  $(ACPICA)/compiler/prparser.l
                $(FLEX) $(LFLAGS) -PPrParser -o$@ $(ACPICA)/compiler/prparser.l

prparserparse.c: $(ACPICA)/compiler/prparser.y
                $(BISON) $(YFLAGS) -pPrParser -o$@ $(ACPICA)/compiler/prparser.y
                $(RM) prparser.y.h; $(MV) prparserparse.h prparser.y.h


%.o:            $(ACPICA)/common/%.c
                $(COMPILE.c) -o $@ $<
                $(POST_PROCESS_O)
```

```
127 %.o:            $(ACPICA)/compiler/%.c
128                 $(COMPILE.c) -o $@ $<
129                 $(POST_PROCESS_O)

131 %.o:            $(ACPICA)/components/debugger/%.c
132                 $(COMPILE.c) -o $@ $<
133                 $(POST_PROCESS_O)

135 %.o:            $(ACPICA)/components/disassembler/%.c
136                 $(COMPILE.c) -o $@ $<
137                 $(POST_PROCESS_O)

139 %.o:            $(ACPICA)/components/dispatcher/%.c
140                 $(COMPILE.c) -o $@ $<
141                 $(POST_PROCESS_O)

143 %.o:            $(ACPICA)/components/executer/%.c
144                 $(COMPILE.c) -o $@ $<
145                 $(POST_PROCESS_O)

147 %.o:            $(ACPICA)/components/namespace/%.c
148                 $(COMPILE.c) -o $@ $<
149                 $(POST_PROCESS_O)

151 %.o:            $(ACPICA)/components/parser/%.c
152                 $(COMPILE.c) -o $@ $<
153                 $(POST_PROCESS_O)

155 %.o:            $(ACPICA)/components/tables/%.c
156                 $(COMPILE.c) -o $@ $<
157                 $(POST_PROCESS_O)

159 %.o:            $(ACPICA)/components/utilities/%.c
160                 $(COMPILE.c) -o $@ $<
161                 $(POST_PROCESS_O)

163 %.o:            $(ACPICA)/os_specific/service_layers/%.c
164                 $(COMPILE.c) -o $@ $<
165                 $(POST_PROCESS_O)

167 include        $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    1345 Thu Dec 26 13:48:10 2013
new/usr/src/cmd/mdb/i86pc/modules/apix/amd64/Makefile
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#

MODULE = apix.so
MDBTGT = kvm

MODSRCS = apix.c apic_common.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../../Makefile.cmd.64
include ../../../../intel/Makefile.amd64
include ../../../Makefile.i86pc
include ../../../../Makefile.module

MODSRCS_DIR = ../../common

CPPFLAGS += -DMP -D_MACHDEP
CPPFLAGS += -I../../../../common
CPPFLAGS += -I../../common
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1305 Thu Dec 26 13:48:10 2013
new/usr/src/cmd/mdb/i86pc/modules/apix/ia32/Makefile
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#

MODULE = apix.so
MDBTGT = kvm

MODSRCS = apix.c apic_common.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../intel/Makefile.ia32
include ../../../Makefile.i86pc
include ../../../../Makefile.module

MODSRCS_DIR = ../../common

CPPFLAGS += -DMP -D_MACHDEP
CPPFLAGS += -I../../../../../common
CPPFLAGS += -I../../common
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1359 Thu Dec 26 13:48:11 2013
new/usr/src/cmd/mdb/i86pc/modules/pcplusmp/amd64/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
  23 #

  25 MODULE = pcplusmp.so
  26 MDBTGT = kvm

  28 MODSRCS = pcplusmp.c apic_common.c intr_common.c

  30 include ../../../../../Makefile.cmd
  31 include ../../../../../Makefile.cmd.64
  32 include ../../../../intel/Makefile.amd64
  33 include ../../../Makefile.i86pc
  34 include ../../../../Makefile.module

  36 MODSRCS_DIR = ../../common

  38 CPPFLAGS += -DMP -D_MACHDEP
  39 CPPFLAGS += -I../../../../common
  40 CPPFLAGS += -I../../common
  41 CPPFLAGS += -I$(SRC)/uts/intel
  42 CPPFLAGS += -I$(SRC)/uts/i86pc
  43 CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1319 Thu Dec 26 13:48:11 2013
new/usr/src/cmd/mdb/i86pc/modules/pcplusmp/ia32/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
  23 #

  25 MODULE = pcplusmp.so
  26 MDBTGT = kvm

  28 MODSRCS = pcplusmp.c apic_common.c intr_common.c

  30 include ../../../../../Makefile.cmd
  31 include ../../../../intel/Makefile.ia32
  32 include ../../../Makefile.i86pc
  33 include ../../../../Makefile.module

  35 MODSRCS_DIR = ../../common

  37 CPPFLAGS += -DMP -D_MACHDEP
  38 CPPFLAGS += -I../../../../common
  39 CPPFLAGS += -I../../common
  40 CPPFLAGS += -I$(SRC)/uts/intel
  41 CPPFLAGS += -I$(SRC)/uts/i86pc
  42 CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1443 Thu Dec 26 13:48:12 2013
new/usr/src/cmd/mdb/i86pc/modules/unix/amd64/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #

  26 MODULE = unix.so
  27 MDBTGT = kvm

  29 MODSRCS = unix.c i86mmu.c

  31 include ../../../../../Makefile.cmd
  32 include ../../../../../Makefile.cmd.64
  33 include ../../../../intel/Makefile.amd64
  34 include ../../../Makefile.i86pc
  35 include ../../../../Makefile.module

  37 CPPFLAGS += -DMP -D_MACHDEP
  38 CPPFLAGS += -I../../../../common
  39 CPPFLAGS += -I$(SRC)/uts/i86pc
  40 CPPFLAGS += -I$(SRC)/uts/intel
  41 CPPFLAGS += -I$(SRC)/common

  43 CERRWARN += -_gcc=-Wno-char-subscripts
  44 CERRWARN += -_gcc=-Wno-parentheses
  45 CERRWARN += -_gcc=-Wno-unused-label
  46 CERRWARN += -_gcc=-Wno-uninitialized
```

```
**********************************************************
    1403 Thu Dec 26 13:48:12 2013
new/usr/src/cmd/mdb/i86pc/modules/unix/ia32/Makefile
PANKOVs restructure
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #

  26 MODULE = unix.so
  27 MDBTGT = kvm

  29 MODSRCS = unix.c i86mmu.c

  31 include ../../../../../Makefile.cmd
  32 include ../../../../intel/Makefile.ia32
  33 include ../../../Makefile.i86pc
  34 include ../../../../Makefile.module

  36 CPPFLAGS += -DMP -D_MACHDEP
  37 CPPFLAGS += -I../../../../common
  38 CPPFLAGS += -I$(SRC)/uts/i86pc
  39 CPPFLAGS += -I$(SRC)/uts/intel
  40 CPPFLAGS += -I$(SRC)/common

  42 CERRWARN += -_gcc=-Wno-char-subscripts
  43 CERRWARN += -_gcc=-Wno-parentheses
  44 CERRWARN += -_gcc=-Wno-unused-label
  45 CERRWARN += -_gcc=-Wno-uninitialized
```

```
**********************************************************
    1355 Thu Dec 26 13:48:12 2013
new/usr/src/cmd/mdb/i86pc/modules/uppc/amd64/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 #ident  "%Z%%M% %I%     %E% SMI"

  26 MODULE = uppc.so
  27 MDBTGT = kvm

  29 MODSRCS = uppc.c intr_common.c

  31 include ../../../../../Makefile.cmd
  32 include ../../../../../Makefile.cmd.64
  33 include ../../../../intel/Makefile.amd64
  34 include ../../../Makefile.i86pc
  35 include ../../../../Makefile.module

  37 MODSRCS_DIR = ../../common

  39 CPPFLAGS += -DMP -D_MACHDEP
  40 CPPFLAGS += -I../../common
  41 CPPFLAGS += -I../../../../common
  42 CPPFLAGS += -I$(SRC)/uts/intel
  43 CPPFLAGS += -I$(SRC)/uts/i86pc
  44 CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1315 Thu Dec 26 13:48:13 2013
new/usr/src/cmd/mdb/i86pc/modules/uppc/ia32/Makefile
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "%Z%%M% %I%     %E% SMI"

MODULE = uppc.so
MDBTGT = kvm

MODSRCS = uppc.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../intel/Makefile.ia32
include ../../../Makefile.i86pc
include ../../../../Makefile.module

MODSRCS_DIR = ../../common

CPPFLAGS += -DMP -D_MACHDEP
CPPFLAGS += -I../../common
CPPFLAGS += -I../../../../../common
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1563 Thu Dec 26 13:48:13 2013
new/usr/src/cmd/mdb/i86xpv/modules/unix/amd64/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #

  26 MODULE = unix.so
  27 MDBTGT = kvm

  29 MODSRCS = unix.c i86mmu.c

  31 include ../../../../../Makefile.cmd
  32 include ../../../../../Makefile.cmd.64
  33 include ../../../../intel/Makefile.amd64
  34 include ../../../Makefile.i86xpv
  35 include ../../../../Makefile.module

  37 MODSRCS_DIR = ../../../../i86pc/modules/unix/

  39 CPPFLAGS += -DMP -D_MACHDEP -D__xpv
  40 CPPFLAGS += -I../../../../common
  41 CPPFLAGS += -I$(SRC)/uts/common
  42 CPPFLAGS += -I$(SRC)/uts/i86xpv
  43 CPPFLAGS += -I$(SRC)/uts/i86pc
  44 CPPFLAGS += -I$(SRC)/uts/intel
  45 CPPFLAGS += -I$(SRC)/common

  47 CERRWARN += -_gcc=-Wno-char-subscripts
  48 CERRWARN += -_gcc=-Wno-parentheses
  49 CERRWARN += -_gcc=-Wno-unused-label
  50 CERRWARN += -_gcc=-Wno-uninitialized
```

```
**********************************************************
    1522 Thu Dec 26 13:48:13 2013
new/usr/src/cmd/mdb/i86xpv/modules/unix/ia32/Makefile
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

MODULE = unix.so
MDBTGT = kvm

MODSRCS = unix.c i86mmu.c

include ../../../../../Makefile.cmd
include ../../../../intel/Makefile.ia32
include ../../../Makefile.i86xpv
include ../../../../Makefile.module

MODSRCS_DIR = ../../../../i86pc/modules/unix

CPPFLAGS += -DMP -D_MACHDEP -D__xpv
CPPFLAGS += -I../../../../common
CPPFLAGS += -I$(SRC)/uts/common
CPPFLAGS += -I$(SRC)/uts/i86xpv
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/common

CERRWARN += -_gcc=-Wno-char-subscripts
CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-unused-label
CERRWARN += -_gcc=-Wno-uninitialized
```

```
**********************************************************
    1474 Thu Dec 26 13:48:14 2013
new/usr/src/cmd/mdb/i86xpv/modules/xpv_psm/amd64/Makefile
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "%Z%%M% %I%     %E% SMI"

MODULE = xpv_psm.so
MDBTGT = kvm

MODSRCS = xpv_psm.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../../Makefile.cmd.64
include ../../../../intel/Makefile.amd64
include ../../../Makefile.i86xpv
include ../../../../Makefile.module

MODSRCS_DIR = ../../../../i86pc/modules/common

CPPFLAGS += -DMP -D_MACHDEP -D__xen
CPPFLAGS += -I../../../../common
CPPFLAGS += -I../../../../i86pc/modules/common
CPPFLAGS += -I$(SRC)/uts/common
CPPFLAGS += -I$(SRC)/uts/i86xpv
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1434 Thu Dec 26 13:48:14 2013
new/usr/src/cmd/mdb/i86xpv/modules/xpv_psm/ia32/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 #ident  "%Z%%M% %I%     %E% SMI"

  26 MODULE = xpv_psm.so
  27 MDBTGT = kvm

  29 MODSRCS = xpv_psm.c intr_common.c

  31 include ../../../../../Makefile.cmd
  32 include ../../../../intel/Makefile.ia32
  33 include ../../../Makefile.i86xpv
  34 include ../../../../Makefile.module

  36 MODSRCS_DIR = ../../../../i86pc/modules/common

  38 CPPFLAGS += -DMP -D_MACHDEP -D__xpv
  39 CPPFLAGS += -I../../../../common
  40 CPPFLAGS += -I../../../../i86pc/modules/common
  41 CPPFLAGS += -I$(SRC)/uts/common
  42 CPPFLAGS += -I$(SRC)/uts/i86xpv
  43 CPPFLAGS += -I$(SRC)/uts/i86pc
  44 CPPFLAGS += -I$(SRC)/uts/intel
  45 CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1476 Thu Dec 26 13:48:14 2013
new/usr/src/cmd/mdb/i86xpv/modules/xpv_uppc/amd64/Makefile
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "%Z%%M% %I%     %E% SMI"

MODULE = xpv_uppc.so
MDBTGT = kvm

MODSRCS = xpv_uppc.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../../Makefile.cmd.64
include ../../../../intel/Makefile.amd64
include ../../../Makefile.i86xpv
include ../../../../Makefile.module

MODSRCS_DIR = ../../../../i86pc/modules/common

CPPFLAGS += -DMP -D_MACHDEP -D__xen
CPPFLAGS += -I../../../../common
CPPFLAGS += -I../../../../i86pc/modules/common
CPPFLAGS += -I$(SRC)/uts/common
CPPFLAGS += -I$(SRC)/uts/i86xpv
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/common
```

```
**********************************************************
    1436 Thu Dec 26 13:48:15 2013
new/usr/src/cmd/mdb/i86xpv/modules/xpv_uppc/ia32/Makefile
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "%Z%%M% %I%     %E% SMI"

MODULE = xpv_uppc.so
MDBTGT = kvm

MODSRCS = xpv_uppc.c intr_common.c

include ../../../../../Makefile.cmd
include ../../../../intel/Makefile.ia32
include ../../../Makefile.i86xpv
include ../../../../Makefile.module

MODSRCS_DIR = ../../../../i86pc/modules/common

CPPFLAGS += -DMP -D_MACHDEP -D__xpv
CPPFLAGS += -I../../../../common
CPPFLAGS += -I../../../../i86pc/modules/common
CPPFLAGS += -I$(SRC)/uts/common
CPPFLAGS += -I$(SRC)/uts/i86xpv
CPPFLAGS += -I$(SRC)/uts/i86pc
CPPFLAGS += -I$(SRC)/uts/intel
CPPFLAGS += -I$(SRC)/common
```

****
168 Thu Dec 26 13:48:15 2013
new/usr/src/common/acpica/README.txt
PANKOVs restructure
****

```
   1 The acpica module includes Intel ACPI CA source code drops. No changes are made
   2 to Intel-provided source code.

   4 Latest version is at https://www.acpica.org/downloads/.
```

****
1872 Thu Dec 26 13:48:15 2013
new/usr/src/common/acpica/THIRDPARTYLICENSE
acpica-unix2-20130823
****

```
 1  * Copyright (C) 2000 - 2013, Intel Corp.
 1  * Copyright (C) 2000 - 2011, Intel Corp.
 2  * All rights reserved.
 3  *
 4  * Redistribution and use in source and binary forms, with or without
 5  * modification, are permitted provided that the following conditions
 6  * are met:
 7  * 1. Redistributions of source code must retain the above copyright
 8  *    notice, this list of conditions, and the following disclaimer,
 9  *    without modification.
10  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
11  *    substantially similar to the "NO WARRANTY" disclaimer below
12  *    ("Disclaimer") and any redistribution must be conditioned upon
13  *    including a substantially similar Disclaimer requirement for further
14  *    binary redistribution.
15  * 3. Neither the names of the above-listed copyright holders nor the names
16  *    of any contributors may be used to endorse or promote products derived
17  *    from this software without specific prior written permission.
18  *
19  * Alternatively, this software may be distributed under the terms of the
20  * GNU General Public License ("GPL") version 2 as published by the Free
21  * Software Foundation.
22  *
23  * NO WARRANTY
24  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
25  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
26  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
27  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
28  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
29  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
30  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
31  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
32  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
33  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
34  * POSSIBILITY OF SUCH DAMAGES.
```

```
**********************************************************
  562098 Thu Dec 26 13:48:16 2013
new/usr/src/common/acpica/changes.txt
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 ----------------------------------------
   2 18 December 2013. Summary of changes for version 20131218:
   2 27 May 2011. Summary of changes for version 20110527:

   4 Global note: The ACPI 5.0A specification was released this month. There
   5 are no changes needed for ACPICA since this release of ACPI is an
   6 errata/clarification release. The specification is available at
   7 acpi.info.
   4 This release is available at www.acpica.org/downloads

  10 1) ACPICA kernel-resident subsystem:

  12 Added validation of the XSDT root table if it is present. Some older
  13 platforms contain an XSDT that is ill-formed or otherwise invalid (such
  14 as containing some or all entries that are NULL pointers). This change
  15 adds a new function to validate the XSDT before actually using it. If the
  16 XSDT is found to be invalid, ACPICA will now automatically fall back to
  17 using the RSDT instead. Original implementation by Zhao Yakui. Ported to
  18 ACPICA and enhanced by Lv Zheng and Bob Moore.

  20 Added a runtime option to ignore the XSDT and force the use of the RSDT.
  21 This change adds a runtime option that will force ACPICA to use the RSDT
  22 instead of the XSDT (AcpiGbl_DoNotUseXsdt). Although the ACPI spec
  23 requires that an XSDT be used instead of the RSDT, the XSDT has been
  24 found to be corrupt or ill-formed on some machines. Lv Zheng.

  26 Added a runtime option to favor 32-bit FADT register addresses over the
  27 64-bit addresses. This change adds an option to favor 32-bit FADT
  28 addresses when there is a conflict between the 32-bit and 64-bit versions
  29 of the same register. The default behavior is to use the 64-bit version
  30 in accordance with the ACPI specification. This can now be overridden via
  31 the AcpiGbl_Use32BitFadtAddresses flag. ACPICA BZ 885. Lv Zheng.

  33 During the change above, the internal "Convert FADT" and "Verify FADT"
  34 functions have been merged to simplify the code, making it easier to
  35 understand and maintain. ACPICA BZ 933.

  37 Improve exception reporting and handling for GPE block installation.
  38 Return an actual status from AcpiEvGetGpeXruptBlock and don't clobber the
  39 status when exiting AcpiEvInstallGpeBlock. ACPICA BZ 1019.

  41 Added helper macros to extract bus/segment numbers from the HEST table.
  42 This change adds two macros to extract the encoded bus and segment
  43 numbers from the HEST Bus field - ACPI_HEST_BUS and ACPI_HEST_SEGMENT.
  44 Betty Dall <betty.dall@hp.com>

  46 Removed the unused ACPI_FREE_BUFFER macro. This macro is no longer used
  47 by ACPICA. It is not a public macro, so it should have no effect on
  48 existing OSV code. Lv Zheng.

  50 Example Code and Data Size: These are the sizes for the OS-independent
  51 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
  52 debug version of the code includes the debug output trace mechanism and
  53 has a much larger code and data size.

  55   Current Release:
```



```
  56     Non-Debug Version:  96.1K Code, 27.0K Data, 123.1K Total
  57     Debug Version:     185.6K Code, 77.3K Data, 262.9K Total
  58   Previous Release:
  59     Non-Debug Version:  95.9K Code, 27.0K Data, 122.9K Total
  60     Debug Version:     185.1K Code, 77.2K Data, 262.3K Total


  63 2) iASL Compiler/Disassembler and Tools:

  65 Disassembler: Improved pathname support for emitted External()
  66 statements. This change adds full pathname support for external names
  67 that have been resolved internally by the inclusion of additional ACPI
  68 tables (via the iASL -e option). Without this change, the disassembler
  69 can emit multiple externals for the same object, or it become confused
  70 when the Scope() operator is used on an external object. Overall, greatly
  71 improves the ability to actually recompile the emitted ASL code when
  72 objects a referenced across multiple ACPI tables. Reported by Michael
  73 Tsirkin (mst@redhat.com).

  75 Tests/ASLTS: Updated functional control suite to execute with no errors.
  76 David Box. Fixed several errors related to the testing of the interpreter
  77 slack mode. Lv Zheng.

  79 iASL: Added support to detect names that are declared within a control
  80 method, but are unused (these are temporary names that are only valid
  81 during the time the method is executing). A remark is issued for these
  82 cases. ACPICA BZ 1022.

  84 iASL: Added full support for the DBG2 table. Adds full disassembler,
  85 table compiler, and template generator support for the DBG2 table (Debug
  86 Port 2 table).

  88 iASL: Added full support for the PCCT table, update the table definition.
  89 Updates the PCCT table definition in the actbl3.h header and adds table
  90 compiler and template generator support.

  92 iASL: Added an option to emit only error messages (no warnings/remarks).
  93 The -ve option will enable only error messages, warnings and remarks are
  94 suppressed. This can simplify debugging when only the errors are
  95 important, such as when an ACPI table is disassembled and there are many
  96 warnings and remarks -- but only the actual errors are of real interest.

  98 Example ACPICA code (source/tools/examples): Updated the example code so
  99 that it builds to an actual working program, not just example code. Added
 100 ACPI tables and execution of an example control method in the DSDT. Added
 101 makefile support for Unix generation.

 103 ----------------------------------------
 104 15 November 2013. Summary of changes for version 20131115:

 106 This release is available at https://acpica.org/downloads


 109 1) ACPICA kernel-resident subsystem:

 111 Resource Manager: Fixed loop termination for the "get AML length"
 112 function. The loop previously had an error termination on a NULL resource
 113 pointer, which can never happen since the loop simply increments a valid
 114 resource pointer. This fix changes the loop to terminate with an error on
 115 an invalid end-of-buffer condition. The problem can be seen as an
 116 infinite loop by callers to AcpiSetCurrentResources with an invalid or
 117 corrupted resource descriptor, or a resource descriptor that is missing
 118 an END_TAG descriptor. Reported by Dan Carpenter
 119 <dan.carpenter@oracle.com>. Lv Zheng, Bob Moore.

 121 Table unload and ACPICA termination: Delete all attached data objects
```

during namespace node deletion. This fix updates namespace node deletion to delete the entire list of attached objects (attached via AcpiAttachObject) instead of just one of the attached items. ACPICA BZ 1024. Tomasz Nowicki (tomasz.nowicki@linaro.org).

ACPICA termination: Added support to delete all objects attached to the root namespace node. This fix deletes any and all objects that have been attached to the root node via AcpiAttachData. Previously, none of these objects were deleted. Reported by Tomasz Nowicki. ACPICA BZ 1026.

Debug output: Do not emit the function nesting level for the in-kernel build. The nesting level is really only useful during a single-thread execution. Therefore, only enable this output for the AcpiExec utility. Also, only emit the thread ID when executing under AcpiExec (Context switches are still always detected and a message is emitted). ACPICA BZ 972.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Current Release:
    Non-Debug Version:  95.9K Code, 27.0K Data, 122.9K Total
    Debug Version:     185.1K Code, 77.2K Data, 262.3K Total
  Previous Release:
    Non-Debug Version:  95.8K Code, 27.0K Data, 122.8K Total
    Debug Version:     185.2K Code, 77.2K Data, 262.4K Total
```

2) iASL Compiler/Disassembler and Tools:

AcpiExec/Unix-OSL: Use <termios.h> instead of <termio.h>. This is the correct portable POSIX header for terminal control functions.

Disassembler: Fixed control method invocation issues related to the use of the CondRefOf() operator. The problem is seen in the disassembly where control method invocations may not be disassembled properly if the control method name has been used previously as an argument to CondRefOf. The solution is to not attempt to emit an external declaration for the CondRefOf target (it is not necessary in the first place). This prevents disassembler object type confusion. ACPICA BZ 988.

Unix Makefiles: Added an option to disable compiler optimizations and the _FORTIFY_SOURCE flag. Some older compilers have problems compiling ACPICA with optimizations (reportedly, gcc 4.4 for example). This change adds a command line option for make (NOOPT) that disables all compiler optimizations and the _FORTIFY_SOURCE compiler flag. The default optimization is -O2 with the _FORTIFY_SOURCE flag specified. ACPICA BZ 1034. Lv Zheng, Bob Moore.

Tests/ASLTS: Added options to specify individual test cases and modes. This allows testers running aslts.sh to optionally specify individual test modes and test cases. Also added an option to disable the forced generation of the ACPICA tools from source if desired. Lv Zheng.

---

27 September 2013. Summary of changes for version 20130927:

This release is available at https://acpica.org/downloads

1) ACPICA kernel-resident subsystem:

Fixed a problem with store operations to reference objects. This change fixes a problem where a Store operation to an ArgX object that contained a reference to a field object did not complete the automatic dereference and then write to the actual field object. Instead, the object type of the field object was inadvertently changed to match the type of the source operand. The new behavior will actually write to the field object (buffer field or field unit), thus matching the correct ACPI-defined behavior.

Implemented support to allow the host to redefine individual OSL prototypes. This change enables the host to redefine OSL prototypes found in the acpiosxf.h file. This allows the host to implement OSL interfaces with a macro or inlined function. Further, it allows the host to add any additional required modifiers such as __iomem, __init, __exit, etc., as necessary on a per-interface basis. Enables maximum flexibility for the OSL interfaces. Lv Zheng.

Hardcoded the access width for the FADT-defined reset register. The ACPI specification requires the reset register width to be 8 bits. ACPICA now hardcodes the width to 8 and ignores the FADT width value. This provides compatibility with other ACPI implementations that have allowed BIOS code with bad register width values to go unnoticed. Matthew Garett, Bob Moore,
Lv Zheng.

Changed the position/use of the ACPI_PRINTF_LIKE macro. This macro is used in the OSL header (acpiosxf). The change modifies the position of this macro in each instance where it is used (AcpiDebugPrint, etc.) to avoid build issues if the OSL defines the implementation of the interface to be an inline stub function. Lv Zheng.

Deployed a new macro ACPI_EXPORT_SYMBOL_INIT for the main ACPICA initialization interfaces. This change adds a new macro for the main init and terminate external interfaces in order to support hosts that require additional or different processing for these functions. Changed from ACPI_EXPORT_SYMBOL to ACPI_EXPORT_SYMBOL_INIT for these functions. Lv Zheng, Bob Moore.

Cleaned up the memory allocation macros for configurability. In the common case, the ACPI_ALLOCATE and related macros now resolve directly to their respective AcpiOs* OSL interfaces. Two options:
1) The ACPI_ALLOCATE_ZEROED macro uses a simple local implementation by default, unless overridden by the USE_NATIVE_ALLOCATE_ZEROED define.
2) For AcpiExec (and for debugging), the macros can optionally be resolved to the local ACPICA interfaces that track each allocation (local tracking is used to immediately detect memory leaks).
Lv Zheng.

Simplified the configuration for ACPI_REDUCED_HARDWARE. Allows the kernel to predefine this macro to either TRUE or FALSE during the system build.

Replaced __FUNCTION_ with __func__ in the gcc-specific header.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Current Release:
    Non-Debug Version:  95.8K Code, 27.0K Data, 122.8K Total
    Debug Version:     185.2K Code, 77.2K Data, 262.4K Total
  Previous Release:
    Non-Debug Version:  96.7K Code, 27.1K Data, 123.9K Total
    Debug Version:     184.4K Code, 76.8K Data, 261.2K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented wildcard support for the -e option. This simplifies use when there are many SSDTs that must be included to resolve external method declarations. ACPICA BZ 1041. Example:

```
iasl -e ssdt*.dat -d dsdt.dat
```

AcpiExec: Add history/line-editing for Unix/Linux systems. This change adds a portable module that implements full history and limited line editing for Unix and Linux systems. It does not use readline() due to portability issues. Instead it uses the POSIX termio interface to put the terminal in raw input mode so that the various special keys can be trapped (such as up/down-arrow for history support and left/right-arrow for line editing). Uses the existing debugger history mechanism. ACPICA BZ 1036.

AcpiXtract: Add support to handle (ignore) "empty" lines containing only one or more spaces. This provides compatible with early or different versions of the AcpiDump utility. ACPICA BZ 1044.

AcpiDump: Do not ignore tables that contain only an ACPI table header. Apparently, some BIOSs create SSDTs that contain an ACPI table header but no other data. This change adds support to dump these tables. Any tables shorter than the length of an ACPI table header remain in error (an error message is emitted). Reported by Yi Li.

Debugger: Echo actual command along with the "unknown command" message.

---

23 August 2013. Summary of changes for version 20130823:

1) ACPICA kernel-resident subsystem:

Implemented support for host-installed System Control Interrupt (SCI) handlers. Certain ACPI functionality requires the host to handle raw SCIs. For example, the "SCI Doorbell" that is defined for memory power state support requires the host device driver to handle SCIs to examine if the doorbell has been activated. Multiple SCI handlers can be installed to allow for future expansion. New external interfaces are AcpiInstallSciHandler, AcpiRemoveSciHandler; see the ACPICA reference for details. Lv Zheng, Bob Moore. ACPICA BZ 1032.

Operation region support: Never locally free the handler "context" pointer. This change removes some dangerous code that attempts to free the handler context pointer in some (rare) circumstances. The owner of the handler owns this pointer and the ACPICA code should never touch it. Although not seen to be an issue in any kernel, it did show up as a problem (fault) under AcpiExec. Also, set the internal storage field for the context pointer to zero when the region is deactivated, simply for sanity. David Box. ACPICA BZ 1039.

AcpiRead: On error, do not modify the return value target location. If an error happens in the middle of a split 32/32 64-bit I/O operation, do not modify the target of the return value pointer. Makes the code consistent with the rest of ACPICA. Bjorn Helgaas.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Current Release:
    Non-Debug Version:  96.7K Code, 27.1K Data, 123.9K Total
```



```
    Debug Version:     184.4K Code, 76.8K Data, 261.2K Total
  Previous Release:
    Non-Debug Version:  96.2K Code, 27.1K Data, 123.3K Total
    Debug Version:     185.4K Code, 77.1K Data, 262.5K Total
```

2) iASL Compiler/Disassembler and Tools:

AcpiDump: Implemented several new features and fixed some problems:
1) Added support to dump the RSDP, RSDT, and XSDT tables.
2) Added support for multiple table instances (SSDT, UEFI).
3) Added option to dump "customized" (overridden) tables (-c).
4) Fixed a problem where some table filenames were improperly constructed.
5) Improved some error messages, removed some unnecessary messages.

iASL: Implemented additional support for disassembly of ACPI tables that contain invocations of external control methods. The -fe<file> option allows the import of a file that specifies the external methods along with the required number of arguments for each -- allowing for the correct disassembly of the table. This is a workaround for a limitation of AML code where the disassembler often cannot determine the number of arguments required for an external control method and generates incorrect ASL code. See the iASL reference for details. ACPICA BZ 1030.

Debugger: Implemented a new command (paths) that displays the full pathnames (namepaths) and object types of all objects in the namespace. This is an alternative to the namespace command.

Debugger: Implemented a new command (sci) that invokes the SCI dispatch mechanism and any installed handlers.

iASL: Fixed a possible segfault for "too many parent prefixes" condition. This can occur if there are too many parent prefixes in a namepath (for example, ^^^^^^PCI0.ECRD). ACPICA BZ 1035.

Application OSLs: Set the return value for the PCI read functions. These functions simply return AE_OK, but should set the return value to zero also. This change implements this. ACPICA BZ 1038.

Debugger: Prevent possible command line buffer overflow. Increase the size of a couple of the debugger line buffers, and ensure that overflow cannot happen. ACPICA BZ 1037.

iASL: Changed to abort immediately on serious errors during the parsing phase. Due to the nature of ASL, there is no point in attempting to compile these types of errors, and they typically end up causing a cascade of hundreds of errors which obscure the original problem.

---

25 July 2013. Summary of changes for version 20130725:

1) ACPICA kernel-resident subsystem:

Fixed a problem with the DerefOf operator where references to FieldUnits and BufferFields incorrectly returned the parent object, not the actual value of the object. After this change, a dereference of a FieldUnit reference results in a read operation on the field to get the value, and likewise, the appropriate BufferField value is extracted from the target buffer.

Fixed a problem where the _WAK method could cause a fault under these circumstances: 1) Interpreter slack mode was not enabled, and 2) the _WAK method returned no value. The problem is rarely seen because most kernels run ACPICA in slack mode.

For the DerefOf operator, a fatal error now results if an attempt is made
to dereference a reference (created by the Index operator) to a NULL
package element. Provides compatibility with other ACPI implementations,
and this behavior will be added to a future version of the ACPI
specification.

The ACPI Power Management Timer (defined in the FADT) is now optional.
This provides compatibility with other ACPI implementations and will
appear in the next version of the ACPI specification. If there is no PM
Timer on the platform, AcpiGetTimer returns AE_SUPPORT. An address of
zero in the FADT indicates no PM timer.

Implemented a new interface for _OSI support, AcpiUpdateInterfaces. This
allows the host to globally enable/disable all vendor strings, all
feature strings, or both. Intended to be primarily used for debugging
purposes only. Lv Zheng.

Expose the collected _OSI data to the host via a global variable. This
data tracks the highest level vendor ID that has been invoked by the BIOS
so that the host (and potentially ACPICA itself) can change behaviors
based upon the age of the BIOS.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

```
  Current Release:
    Non-Debug Version:  96.2K Code, 27.1K Data, 123.3K Total
    Debug Version:     184.4K Code, 76.8K Data, 261.2K Total
  Previous Release:
    Non-Debug Version:  95.9K Code, 26.9K Data, 122.8K Total
    Debug Version:     184.1K Code, 76.7K Data, 260.8K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Created the following enhancements for the -so option (create
offset table):
1)Add offsets for the last nameseg in each namepath for every supported
object type
2)Add support for Processor, Device, Thermal Zone, and Scope objects
3)Add the actual AML opcode for the parent object of every supported
object type
4)Add support for the ZERO/ONE/ONES AML opcodes for integer objects

Disassembler: Emit all unresolved external symbols in a single block.
These are external references to control methods that could not be
resolved, and thus, the disassembler had to make a guess at the number of
arguments to parse.

iASL: The argument to the -T option (create table template) is now
optional. If not specified, the default table is a DSDT, typically the
most common case.

---

26 June 2013. Summary of changes for version 20130626:

1) ACPICA kernel-resident subsystem:

Fixed an issue with runtime repair of the _CST object. Null or invalid
elements were not always removed properly. Lv Zheng.

Removed an arbitrary restriction of 256 GPEs per GPE block (such as the
FADT-defined GPE0 and GPE1). For GPE0, GPE1, and each GPE Block Device,
the maximum number of GPEs is 1016. Use of multiple GPE block devices

makes the system-wide number of GPEs essentially unlimited.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

```
  Current Release:
    Non-Debug Version:  95.9K Code, 26.9K Data, 122.8K Total
    Debug Version:     184.1K Code, 76.7K Data, 260.8K Total
  Previous Release:
    Non-Debug Version:  96.0K Code, 27.0K Data, 123.0K Total
    Debug Version:     184.1K Code, 76.8K Data, 260.9K Total
```

2) iASL Compiler/Disassembler and Tools:

Portable AcpiDump: Implemented full support for the Linux and FreeBSD
hosts. Now supports Linux, FreeBSD, and Windows.

Disassembler: Added some missing types for the HEST and EINJ tables: "Set
Error Type With Address", "CMCI", "MCE", and "Flush Cacheline".

iASL/Preprocessor: Implemented full support for nested
#if/#else/#elif/#endif blocks. Allows arbitrary depth of nested blocks.

Disassembler: Expanded maximum output string length to 64K. Was 256 bytes
max. The original purpose of this constraint was to limit the amount of
debug output. However, the string function in question (UtPrintString) is
now used for the disassembler also, where 256 bytes is insufficient.
Reported by RehabMan@GitHub.

iASL/DataTables: Fixed some problems and issues with compilation of DMAR
tables. ACPICA BZ 999. Lv Zheng.

iASL: Fixed a couple of error exit issues that could result in a "Could
not delete <file>" message during ASL compilation.

AcpiDump: Allow "FADT" and "MADT" as valid table signatures, even though
the actual signatures for these tables are "FACP" and "APIC",
respectively.

AcpiDump: Added support for multiple UEFI tables. Only SSDT and UEFI
tables are allowed to have multiple instances.

---

17 May 2013. Summary of changes for version 20130517:

1) ACPICA kernel-resident subsystem:

Fixed a regression introduced in version 20130328 for _INI methods. This
change fixes a problem introduced in 20130328 where _INI methods are no
longer executed properly because of a memory block that was not
initialized correctly. ACPICA BZ 1016. Tomasz Nowicki
<tomasz.nowicki@linaro.org>.

Fixed a possible problem with the new extended sleep registers in the
ACPI
5.0 FADT. Do not use these registers (even if populated) unless the HW-
reduced bit is set in the FADT (as per the ACPI specification). ACPICA BZ
1020. Lv Zheng.

Implemented return value repair code for _CST predefined objects: Sort
the
list and detect/remove invalid entries. ACPICA BZ 890. Lv Zheng.

Implemented a debug-only option to disable loading of SSDTs from the
RSDT/XSDT during ACPICA initialization. This can be useful for debugging
ACPI problems on some machines. Set AcpiGbl_DisableSsdtTableLoad in
acglobal.h - ACPICA BZ 1005. Lv Zheng.

Fixed some issues in the ACPICA initialization and termination code:
Tomasz Nowicki <tomasz.nowicki@linaro.org>
1) Clear events initialized flag upon event component termination. ACPICA
BZ 1013.
2) Fixed a possible memory leak in GPE init error path. ACPICA BZ 1018.
3) Delete global lock pending lock during termination. ACPICA BZ 1012.
4) Clear debug buffer global on termination to prevent possible multiple
delete. ACPICA BZ 1010.

Standardized all switch() blocks across the entire source base. After
many
years, different formatting for switch() had crept in. This change makes
the formatting of every switch block identical. ACPICA BZ 997. Chao Guan.

Split some files to enhance ACPICA modularity and configurability:
1) Split buffer dump routines into utilities/utbuffer.c
2) Split internal error message routines into utilities/uterror.c
3) Split table print utilities into tables/tbprint.c
4) Split iASL command-line option processing into asloptions.c

Makefile enhancements:
1) Support for all new files above.
2) Abort make on errors from any subcomponent. Chao Guan.
3) Add build support for Apple Mac OS X. Liang Qi.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

  Current Release:
    Non-Debug Version: 96.0K Code, 27.0K Data, 123.0K Total
    Debug Version: 184.1K Code, 76.8K Data, 260.9K Total
  Previous Release:
    Non-Debug Version: 95.6K Code, 26.8K Data, 122.4K Total
    Debug Version: 183.5K Code, 76.6K Data, 260.1K Total

2) iASL Compiler/Disassembler and Tools:

New utility: Implemented an easily portable version of the acpidump
utility to extract ACPI tables from the system (or a file) in an ASCII
hex
dump format. The top-level code implements the various command line
options, file I/O, and table dump routines. To port to a new host, only
three functions need to be implemented to get tables -- since this
functionality is OS-dependent. See the tools/acpidump/apmain.c module and
the ACPICA reference for porting instructions. ACPICA BZ 859. Notes:
1) The Windows version obtains the ACPI tables from the Registry.
2) The Linux version is under development.
3) Other hosts - If an OS-dependent module is submitted, it will be
distributed with ACPICA.

iASL: Fixed a regression for -D preprocessor option (define symbol). A
restructuring/change to the initialization sequence caused this option to
no longer work properly.

iASL: Implemented a mechanism to disable specific warnings and remarks.
Adds a new command line option, "-vw <messageid> as well as "#pragma
disable <messageid>". ACPICA BZ 989. Chao Guan, Bob Moore.

iASL: Fix for too-strict package object validation. The package object
validation for return values from the predefined names is a bit too
strict, it does not allow names references within the package (which will
be resolved at runtime.) These types of references cannot be validated at
compile time. This change ignores named references within package objects
for names that return or define static packages.

Debugger: Fixed the 80-character command line limitation for the History
command. Now allows lines of arbitrary length. ACPICA BZ 1000. Chao Guan.

iASL: Added control method and package support for the -so option
(generates AML offset table for BIOS support.)

iASL: issue a remark if a non-serialized method creates named objects. If
a thread blocks within the method for any reason, and another thread
enters the method, the method will fail because an attempt will be made
to
create the same (named) object twice. In this case, issue a remark that
the method should be marked serialized. NOTE: may become a warning later.
ACPICA BZ 909.

----
18 April 2013. Summary of changes for version 20130418:

1) ACPICA kernel-resident subsystem:

Fixed a possible buffer overrun during some rare but specific field unit
read operations. This overrun can only happen if the DSDT version is 1 --
meaning that all AML integers are 32 bits -- and the field length is
between 33 and 55 bits long. During the read, an internal buffer object
is
created for the field unit because the field is larger than an integer
(32
bits). However, in this case, the buffer will be incorrectly written
beyond the end because the buffer length is less than the internal
minimum
of 64 bits (8 bytes) long. The buffer will be either 5, 6, or 7 bytes
long, but a full 8 bytes will be written.

Updated the Embedded Controller "orphan" _REG method support. This refers
to _REG methods under the EC device that have no corresponding operation
region. This is allowed by the ACPI specification. This update removes a
dependency on the existence an ECDT table. It will execute an orphan _REG
method as long as the operation region handler for the EC is installed at
the EC device node and not the namespace root. Rui Zhang (original
update), Bob Moore (update/integrate).

Implemented run-time argument typechecking for all predefined ACPI names
(_STA, _BIF, etc.) This change performs object typechecking on all
incoming arguments for all predefined names executed via
AcpiEvaluateObject. This ensures that ACPI-related device drivers are
passing correct object types as well as the correct number of arguments
(therefore identifying any issues immediately). Also, the ASL/namespace
definition of the predefined name is checked against the ACPI
specification for the proper argument count. Adds one new file,
nsarguments.c

Changed an exception code for the ASL UnLoad() operator. Changed the
exception code for the case where the input DdbHandle is invalid, from
AE_BAD_PARAMETER to the more appropriate AE_AML_OPERAND_TYPE.

Unix/Linux makefiles: Removed the use of the -O2 optimization flag in the
global makefile. The use of this flag causes compiler errors on earlier
versions of GCC, so it has been removed for compatibility.

Miscellaneous cleanup:

```
1) Removed some unused/obsolete macros
2) Fixed a possible memory leak in the _OSI support
3) Removed an unused variable in the predefined name support
4) Windows OSL: remove obsolete reference to a memory list field

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

  Current Release:
    Non-Debug Version:  95.2K Code, 26.4K Data, 121.6K Total
    Debug Version:     183.0K Code, 76.0K Data, 259.0K Total
  Previous Release:
    Non-Debug Version:  95.6K Code, 26.8K Data, 122.4K Total
    Debug Version:     183.5K Code, 76.6K Data, 260.1K Total


2) iASL Compiler/Disassembler and Tools:

AcpiExec: Added installation of a handler for the SystemCMOS address
space. This prevents control method abort if a method accesses this
space.

AcpiExec: Added support for multiple EC devices, and now install EC
operation region handler(s) at the actual EC device instead of the
namespace root. This reflects the typical behavior of host operating
systems.

AcpiExec: Updated to ensure that all operation region handlers are
installed before the _REG methods are executed. This prevents a _REG
method from aborting if it accesses an address space has no handler.
AcpiExec installs a handler for every possible address space.

Debugger: Enhanced the "handlers" command to display non-root handlers.
This change enhances the handlers command to display handlers associated
with individual devices throughout the namespace, in addition to the
currently supported display of handlers associated with the root
namespace
node.

ASL Test Suite: Several test suite errors have been identified and
resolved, reducing the total error count during execution. Chao Guan.

---------------------------------------
28 March 2013. Summary of changes for version 20130328:

1) ACPICA kernel-resident subsystem:

Fixed several possible race conditions with the internal object reference
counting mechanism. Some of the external ACPICA interfaces update object
reference counts without holding the interpreter or namespace lock. This
change adds a spinlock to protect reference count updates on the internal
ACPICA objects. Reported by and with assistance from Andriy Gapon
(avg@FreeBSD.org).

FADT support: Removed an extraneous warning for very large GPE register
sets. This change removes a size mismatch warning if the legacy length
field for a GPE register set is larger than the 64-bit GAS structure can
accommodate. GPE register sets can be larger than the 255-bit width
limitation of the GAS structure. Linn Crosetto (linn@hp.com).

_OSI Support: handle any errors from AcpiOsAcquireMutex. Check for error
return from this interface. Handles a possible timeout case if
ACPI_WAIT_FOREVER is modified by the host to be a value less than
"forever". Jung-uk Kim.
```



```
Predefined name support: Add allowed/required argument type information
to
the master predefined info table. This change adds the infrastructure to
enable typechecking on incoming arguments for all predefined
methods/objects. It does not actually contain the code that will fully
utilize this information, this is still under development. Also condenses
some duplicate code for the predefined names into a new module,
utilities/utpredef.c

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

  Previous Release:
    Non-Debug Version:  95.0K Code, 25.9K Data, 120.9K Total
    Debug Version:     182.9K Code, 75.6K Data, 258.5K Total
  Current Release:
    Non-Debug Version:  95.2K Code, 26.4K Data, 121.6K Total
    Debug Version:     183.0K Code, 76.0K Data, 259.0K Total


2) iASL Compiler/Disassembler and Tools:

iASL: Implemented a new option to simplify the development of ACPI-
related
BIOS code. Adds support for a new "offset table" output file. The -so
option will create a C table containing the AML table offsets of various
named objects in the namespace so that BIOS code can modify them easily
at
boot time. This can simplify BIOS runtime code by eliminating expensive
searches for "magic values", enhancing boot times and adding greater
reliability. With assistance from Lee Hamel.

iASL: Allow additional predefined names to return zero-length packages.
Now, all predefined names that are defined by the ACPI specification to
return a "variable-length package of packages" are allowed to return a
zero length top-level package. This allows the BIOS to tell the host that
the requested feature is not supported, and supports existing BIOS/ASL
code and practices.

iASL: Changed the "result not used" warning to an error. This is the case
where an ASL operator is effectively a NOOP because the result of the
operation is not stored anywhere. For example:
    Add (4, Local0)
There is no target (missing 3rd argument), nor is the function return
value used. This is potentially a very serious problem -- since the code
was probably intended to do something, but for whatever reason, the value
was not stored. Therefore, this issue has been upgraded from a warning to
an error.

AcpiHelp: Added allowable/required argument types to the predefined names
info display. This feature utilizes the recent update to the predefined
names table (above).

---------------------------------------
14 February 2013. Summary of changes for version 20130214:

1) ACPICA Kernel-resident Subsystem:

Fixed a possible regression on some hosts: Reinstated the safe return
macros (return_ACPI_STATUS, etc.) that ensure that the argument is
evaluated only once. Although these macros are not needed for the ACPICA
code itself, they are often used by ACPI-related host device drivers
where
```

```
the safe feature may be necessary.

Fixed several issues related to the ACPI 5.0 reduced hardware support
(SOC): Now ensure that if the platform declares itself as hardware-
reduced
via the FADT, the following functions become NOOPs (and always return
AE_OK) because ACPI is always enabled by definition on these machines:
  AcpiEnable
  AcpiDisable
  AcpiHwGetMode
  AcpiHwSetMode

Dynamic Object Repair: Implemented additional runtime repairs for
predefined name return values. Both of these repairs can simplify code in
the related device drivers that invoke these methods:
1) For the _STR and _MLS names, automatically repair/convert an ASCII
string to a Unicode buffer.
2) For the _CRS, _PRS, and _DMA names, return a resource descriptor with
a
lone end tag descriptor in the following cases: A Return(0) was executed,
a null buffer was returned, or no object at all was returned (non-slack
mode only). Adds a new file, nsconvert.c
ACPICA BZ 998. Bob Moore, Lv Zheng.

Resource Manager: Added additional code to prevent possible infinite
loops
while traversing corrupted or ill-formed resource template buffers. Check
for zero-length resource descriptors in all code that loops through
resource templates (the length field is used to index through the
template). This change also hardens the external AcpiWalkResources and
AcpiWalkResourceBuffer interfaces.

Local Cache Manager: Enhanced the main data structure to eliminate an
unnecessary mechanism to access the next object in the list. Actually
provides a small performance enhancement for hosts that use the local
ACPICA cache manager. Jung-uk Kim.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

  Previous Release:
    Non-Debug Version:  94.5K Code, 25.4K Data, 119.9K Total
    Debug Version:     182.3K Code, 75.0K Data, 257.3K Total
  Current Release:
    Non-Debug Version:  95.0K Code, 25.9K Data, 120.9K Total
    Debug Version:     182.9K Code, 75.6K Data, 258.5K Total


2) iASL Compiler/Disassembler and Tools:

iASL/Disassembler: Fixed several issues with the definition of the ACPI
5.0 RASF table (RAS Feature Table). This change incorporates late changes
that were made to the ACPI 5.0 specification.

iASL/Disassembler: Added full support for the following new ACPI tables:
  1) The MTMR table (MID Timer Table)
  2) The VRTC table (Virtual Real Time Clock Table).
Includes header file, disassembler, table compiler, and template support
for both tables.

iASL: Implemented compile-time validation of package objects returned by
predefined names. This new feature validates static package objects
returned by the various predefined names defined to return packages. Both
object types and package lengths are validated, for both parent packages
```



```
and sub-packages, if any. The code is similar in structure and behavior
to
the runtime repair mechanism within the AML interpreter and uses the
existing predefined name information table. Adds a new file, aslprepkg.c.
ACPICA BZ 938.

iASL: Implemented auto-detection of binary ACPI tables for disassembly.
This feature detects a binary file with a valid ACPI table header and
invokes the disassembler automatically. Eliminates the need to
specifically invoke the disassembler with the -d option. ACPICA BZ 862.

iASL/Disassembler: Added several warnings for the case where there are
unresolved control methods during the disassembly. This can potentially
cause errors when the output file is compiled, because the disassembler
assumes zero method arguments in these cases (it cannot determine the
actual number of arguments without resolution/definition of the method).

Debugger: Added support to display all resources with a single command.
Invocation of the resources command with no arguments will now display
all
resources within the current namespace.

AcpiHelp: Added descriptive text for each ACPICA exception code displayed
via the -e option.

----------------------------------------
17 January 2013. Summary of changes for version 20130117:

1) ACPICA Kernel-resident Subsystem:

Updated the AcpiGetSleepTypeData interface: Allow the \_Sx methods to
return either 1 or 2 integers. Although the ACPI spec defines the \_Sx
objects to return a package containing one integer, most BIOS code
returns
two integers and the previous code reflects that. However, we also need
to
support BIOS code that actually implements to the ACPI spec, and this
change reflects this.

Fixed two issues with the ACPI_DEBUG_PRINT macros:
1) Added the ACPI_DO_WHILE macro to the main DEBUG_PRINT helper macro for
C compilers that require this support.
2) Renamed the internal ACPI_DEBUG macro to ACPI_DO_DEBUG_PRINT since
ACPI_DEBUG is already used by many of the various hosts.

Updated all ACPICA copyrights and signons to 2013. Added the 2013
copyright to all module headers and signons, including the standard Linux
header. This affects virtually every file in the ACPICA core subsystem,
iASL compiler, all ACPICA utilities, and the test suites.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a much larger code and data size.

  Previous Release:
    Non-Debug Version:  94.5K Code, 25.5K Data, 120.0K Total
    Debug Version:     182.2K Code, 74.9K Data, 257.1K Total
  Current Release:
    Non-Debug Version:  94.5K Code, 25.4K Data, 119.9K Total
    Debug Version:     182.3K Code, 75.0K Data, 257.3K Total


2) iASL Compiler/Disassembler and Tools:

Generic Unix OSL: Use a buffer to eliminate multiple vfprintf()s and
```

prevent a possible fault on some hosts. Some C libraries modify the arg pointer parameter to vfprintf making it difficult to call it twice in the AcpiOsVprintf function. Use a local buffer to workaround this issue. This does not affect the Windows OSL since the Win C library does not modify the arg pointer. Chao Guan, Bob Moore.

iASL: Fixed a possible infinite loop when the maximum error count is reached. If an output file other than the .AML file is specified (such as a listing file), and the maximum number of errors is reached, do not attempt to flush data to the output file(s) as the compiler is aborting. This can cause an infinite loop as the max error count code essentially keeps calling itself.

iASL/Disassembler: Added an option (-in) to ignore NOOP opcodes/operators. Implemented for both the compiler and the disassembler. Often, the NOOP opcode is used as padding for packages that are changed dynamically by the BIOS. When disassembled and recompiled, these NOOPs will cause syntax errors. This option causes the disassembler to ignore all NOOP opcodes (0xA3), and it also causes the compiler to ignore all ASL source code NOOP statements as well.

Debugger: Enhanced the Sleep command to execute all sleep states. This change allows Sleep to be invoked with no arguments and causes the debugger to execute all of the sleep states, 0-5, automatically.

---

20 December 2012. Summary of changes for version 20121220:

1) ACPICA Kernel-resident Subsystem:

Implemented a new interface, AcpiWalkResourceBuffer. This interface is an alternate entry point for AcpiWalkResources and improves the usability of the resource manager by accepting as input a buffer containing the output of either a _CRS, _PRS, or _AEI method. The key functionality is that the input buffer is not deleted by this interface so that it can be used by the host later. See the ACPICA reference for details.

Interpreter: Add a warning if a 64-bit constant appears in a 32-bit table (DSDT version < 2). The constant will be truncated and this warning reflects that behavior.

Resource Manager: Add support for the new ACPI 5.0 wake bit in the IRQ, ExtendedInterrupt, and GpioInt descriptors. This change adds support to both get and set the new wake bit in these descriptors, separately from the existing share bit. Reported by Aaron Lu.

Interpreter: Fix Store() when an implicit conversion is not possible. For example, in the cases such as a store of a string to an existing package object, implement the store as a CopyObject(). This is a small departure from the ACPI specification which states that the control method should be aborted in this case. However, the ASLTS suite depends on this behavior.

Performance improvement for the various FUNCTION_TRACE and DEBUG_PRINT macros: check if debug output is currently enabled as soon as possible to minimize performance impact if debug is in fact not enabled.

Source code restructuring: Cleanup to improve modularity. The following new files have been added: dbconvert.c, evhandler.c, nsprepkg.c, psopinfo.c, psobject.c, rsdumpinfo.c, utstring.c, and utownerid.c. Associated makefiles and project files have been updated.

Changed an exception code for LoadTable operator. For the case where one

of the input strings is too long, change the returned exception code from AE_BAD_PARAMETER to AE_AML_STRING_LIMIT.

Fixed a possible memory leak in dispatcher error path. On error, delete the mutex object created during method mutex creation. Reported by tim.gardner@canonical.com.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  94.3K Code, 25.3K Data, 119.6K Total
    Debug Version:     175.5K Code, 74.5K Data, 250.0K Total
  Current Release:
    Non-Debug Version:  94.5K Code, 25.5K Data, 120.0K Total
    Debug Version:     182.2K Code, 74.9K Data, 257.1K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Disallow a method call as argument to the ObjectType ASL operator. This change tracks an errata to the ACPI 5.0 document. The AML grammar will not allow the interpreter to differentiate between a method and a method invocation when these are used as an argument to the ObjectType operator. The ACPI specification change is to disallow a method invocation (UserTerm) for the ObjectType operator.

Finish support for the TPM2 and CSRT tables in the headers, table compiler, and disassembler.

Unix user-space OSL: Fix a problem with WaitSemaphore where the timeout always expires immediately if the semaphore is not available. The original code was using a relative-time timeout, but sem_timedwait requires the use of an absolute time.

iASL: Added a remark if the Timer() operator is used within a 32-bit table. This operator returns a 64-bit time value that will be truncated within a 32-bit table.

iASL Source code restructuring: Cleanup to improve modularity. The following new files have been added: aslhex.c, aslxref.c, aslnamesp.c, aslmethod.c, and aslfileio.c. Associated makefiles and project files have been updated.

---

14 November 2012. Summary of changes for version 20121114:

1) ACPICA Kernel-resident Subsystem:

Implemented a performance enhancement for ACPI/AML Package objects. This change greatly increases the performance of Package objects within the interpreter. It changes the processing of reference counts for packages by optimizing for the most common case where the package sub-objects are either Integers, Strings, or Buffers. Increases the overall performance of the ASLTS test suite by 1.5X (Increases the Slack Mode performance by 2X.) Chao Guan. ACPICA BZ 943.

Implemented and deployed common macros to extract flag bits from resource descriptors. Improves readability and maintainability of the code. Fixes a problem with the UART serial bus descriptor for the number of data bits flags (was incorrectly 2 bits, should be 3).

Enhanced the ACPI_GETx and ACPI_SETx macros. Improved the implementation of the macros and changed the SETx macros to the style of (destination, source). Also added ACPI_CASTx companion macros. Lv Zheng.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  93.9K Code, 25.2K Data, 119.1K Total
    Debug Version:     175.5K Code, 74.5K Data, 250.0K Total
  Current Release:
    Non-Debug Version:  94.3K Code, 25.3K Data, 119.6K Total
    Debug Version:     175.5K Code, 74.5K Data, 250.0K Total
```

2) iASL Compiler/Disassembler and Tools:

Disassembler: Added the new ACPI 5.0 interrupt sharing flags. This change adds the ShareAndWake and ExclusiveAndWake flags which were added to the Irq, Interrupt, and Gpio resource descriptors in ACPI 5.0. ACPICA BZ 986.

Disassembler: Fixed a problem with external declaration generation. Fixes a problem where an incorrect pathname could be generated for an external declaration if the original reference to the object includes leading carats (^). ACPICA BZ 984.

Debugger: Completed a major update for the Disassemble<method> command. This command was out-of-date and did not properly disassemble control methods that had any reasonable complexity. This fix brings the command up to the same level as the rest of the disassembler. Adds one new file, dmdeferred.c, which is existing code that is now common with the main disassembler and the debugger disassemble command. ACPICA MZ 978.

iASL: Moved the parser entry prototype to avoid a duplicate declaration. Newer versions of Bison emit this prototype, so moved the prototype out of the iASL header to where it is actually used in order to avoid a duplicate declaration.

iASL/Tools: Standardized use of the stream I/O functions:
  1) Ensure check for I/O error after every fopen/fread/fwrite
  2) Ensure proper order of size/count arguments for fread/fwrite
  3) Use test of (Actual != Requested) after all fwrite, and most fread
  4) Standardize I/O error messages
Improves reliability and maintainability of the code. Bob Moore, Lv Zheng.
ACPICA BZ 981.

Disassembler: Prevent duplicate External() statements. During generation of external statements, detect similar pathnames that are actually duplicates such as these:
  External (\ABCD)
  External (ABCD)
Remove all leading '\' characters from pathnames during the external statement generation so that duplicates will be detected and tossed.
ACPICA BZ 985.

Tools: Replace low-level I/O with stream I/O functions. Replace open/read/write/close with the stream I/O equivalents fopen/fread/fwrite/fclose for portability and performance. Lv Zheng, Bob Moore.

AcpiBin: Fix for the dump-to-hex function. Now correctly output the table name header so that AcpiXtract recognizes the output file/table.

iASL: Remove obsolete -2 option flag. Originally intended to force the compiler/disassembler into an ACPI 2.0 mode, this was never implemented and the entire concept is now obsolete.

---

18 October 2012. Summary of changes for version 20121018:

1) ACPICA Kernel-resident Subsystem:

Updated support for the ACPI 5.0 MPST table. Fixes some problems introduced by late changes to the table as it was added to the ACPI 5.0 specification. Includes header, disassembler, and data table compiler support as well as a new version of the MPST template.

AcpiGetObjectInfo: Enhanced the device object support to include the ACPI 5.0 _SUB method. Now calls _SUB in addition to the other PNP-related ID methods: _HID, _CID, and _UID.

Changed ACPI_DEVICE_ID to ACPI_PNP_DEVICE_ID. Also changed ACPI_DEVICE_ID_LIST to ACPI_PNP_DEVICE_ID_LIST. These changes prevent name collisions on hosts that reserve the *_DEVICE_ID (or *DeviceId) names for their various drivers. Affects the AcpiGetObjectInfo external interface, and other internal interfaces as well.

Added and deployed a new macro for ACPI_NAME management: ACPI_MOVE_NAME. This macro resolves to a simple 32-bit move of the 4-character ACPI_NAME on machines that support non-aligned transfers. Optimizes for this case rather than using a strncpy. With assistance from Zheng Lv.

Resource Manager: Small fix for buffer size calculation. Fixed a one byte error in the output buffer calculation. Feng Tang. ACPICA BZ 849.

Added a new debug print message for AML mutex objects that are force-released. At control method termination, any currently acquired mutex objects are force-released. Adds a new debug-only message for each one that is released.

Audited/updated all ACPICA return macros and the function debug depth counter: 1) Ensure that all functions that use the various TRACE macros also use the appropriate ACPICA return macros. 2) Ensure that all normal return statements surround the return expression (value) with parens to ensure consistency across the ACPICA code base. Guan Chao, Tang Feng, Zheng Lv, Bob Moore. ACPICA Bugzilla 972.

Global source code changes/maintenance: All extra lines at the start and end of each source file have been removed for consistency. Also, within comments, all new sentences start with a single space instead of a double space, again for consistency across the code base.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  93.7K Code, 25.3K Data, 119.0K Total
```

```
    Debug Version:     175.0K Code, 74.4K Data, 249.4K Total
  Current Release:
    Non-Debug Version:  93.9K Code, 25.2K Data, 119.1K Total
    Debug Version:     175.5K Code, 74.5K Data, 250.0K Total


2) iASL Compiler/Disassembler and Tools:

AcpiExec: Improved the algorithm used for memory leak/corruption
detection. Added some intelligence to the code that maintains the global
list of allocated memory. The list is now ordered by allocated memory
address, significantly improving performance. When running AcpiExec on
the ASLTS test suite, speed improvements of 3X to 5X are seen, depending
on the platform and/or the environment. Note, this performance
enhancement affects the AcpiExec utility only, not the kernel-resident
ACPICA code.

Enhanced error reporting for invalid AML opcodes and bad ACPI_NAMEs. For
the disassembler, dump the 48 bytes surrounding the invalid opcode. Fix
incorrect table offset reported for invalid opcodes. Report the original
32-bit value for bad ACPI_NAMEs (as well as the repaired name.)

Disassembler: Enhanced the -vt option to emit the binary table data in
hex format to assist with debugging.

Fixed a potential filename buffer overflow in osunixdir.c. Increased the
size of file structure. Colin Ian King.

----------------------------------------
13 September 2012. Summary of changes for version 20120913:


1) ACPICA Kernel-resident Subsystem:

ACPI 5.0: Added two new notify types for the Hardware Error Notification
Structure within the Hardware Error Source Table (HEST) table -- CMCI(5)
and
MCE(6).

Table Manager: Merged/removed duplicate code in the root table resize
functions. One function is external, the other is internal. Lv Zheng,
ACPICA
BZ 846.

Makefiles: Completely removed the obsolete "Linux" makefiles under
acpica/generate/linux. These makefiles are obsolete and have been
replaced
by
the generic unix makefiles under acpica/generate/unix.

Makefiles: Ensure that binary files always copied properly. Minor rule
change
to ensure that the final binary output files are always copied up to the
appropriate binary directory (bin32 or bin64.)

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug
version of the code includes the debug output trace mechanism and has a
much
larger code and data size.

  Previous Release:
    Non-Debug Version:  93.8K Code, 25.3K Data, 119.1K Total
    Debug Version:     175.7K Code, 74.8K Data, 250.5K Total
  Current Release:
```



```
    Non-Debug Version:  93.7K Code, 25.3K Data, 119.0K Total
    Debug Version:     175.0K Code, 74.4K Data, 249.4K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler: Fixed a possible fault during the disassembly of resource
descriptors when a second parse is required because of the invocation of
external control methods within the table. With assistance from
adq@lidskialf.net. ACPICA BZ 976.

iASL: Fixed a namepath optimization problem. An error can occur if the
parse
node that contains the namepath to be optimized does not have a parent
node
that is a named object. This change fixes the problem.

iASL: Fixed a regression where the AML file is not deleted on errors. The
AML
output file should be deleted if there are any errors during the
compiler.
The
only exception is if the -f (force output) option is used. ACPICA BZ 974.

iASL: Added a feature to automatically increase internal line buffer
sizes.
Via realloc(), automatically increase the internal line buffer sizes as
necessary to support very long source code lines. The current version of
the
preprocessor requires a buffer long enough to contain full source code
lines.
This change increases the line buffer(s) if the input lines go beyond the
current buffer size. This eliminates errors that occurred when a source
code
line was longer than the buffer.

iASL: Fixed a problem with constant folding in method declarations. The
SyncLevel term is a ByteConstExpr, and incorrect code would be generated
if a
Type3 opcode was used.

Debugger: Improved command help support. For incorrect argument count,
display
full help for the command. For help command itself, allow an argument to
specify a command.

Test Suites: Several bug fixes for the ASLTS suite reduces the number of
errors during execution of the suite. Guan Chao.

----------------------------------------
16 August 2012. Summary of changes for version 20120816:


1) ACPICA Kernel-resident Subsystem:

Removed all use of the deprecated _GTS and _BFS predefined methods. The
_GTS
(Going To Sleep) and _BFS (Back From Sleep) methods are essentially
deprecated and will probably be removed from the ACPI specification.
Windows
does not invoke them, and reportedly never will. The final nail in the
coffin
is that the ACPI specification states that these methods must be run with
interrupts off, which is not going to happen in a kernel interpreter.
Note:
Linux has removed all use of the methods also. It was discovered that
```

invoking these functions caused failures on some machines, probably because
they were never tested since Windows does not call them. Affects two external
interfaces, AcpiEnterSleepState and AcpiLeaveSleepStatePrep. Tang Feng.
ACPICA BZ 969.

Implemented support for complex bit-packed buffers returned from the _PLD
(Physical Location of Device) predefined method. Adds a new external
interface, AcpiDecodePldBuffer that parses the buffer into a more usable C
structure. Note: C Bitfields cannot be used for this type of predefined
structure since the memory layout of individual bitfields is not defined by
the C language. In addition, there are endian concerns where a compiler will
change the bitfield ordering based on the machine type. The new ACPICA
interface eliminates these issues, and should be called after _PLD is
executed. ACPICA BZ 954.

Implemented a change to allow a scope change to root (via "Scope (\)") during
execution of module-level ASL code (code that is executed at table load
time.) Lin Ming.

Added the Windows8/Server2012 string for the _OSI method. This change adds a
new _OSI string, "Windows 2012" for both Windows 8 and Windows Server 2012.

Added header support for the new ACPI tables DBG2 (Debug Port Table Type 2)
and CSRT (Core System Resource Table).

Added struct header support for the _FDE, _GRT, _GTM, and _SRT predefined
names. This simplifies access to the buffers returned by these predefined
names. Adds a new file, include/acbuffer.h. ACPICA BZ 956.

GPE support: Removed an extraneous parameter from the various low-level
internal GPE functions. Tang Feng.

Removed the linux makefiles from the unix packages. The generate/linux
makefiles are obsolete and have been removed from the unix tarball release
packages. The replacement makefiles are under generate/unix, and there is a
top-level makefile under the main acpica directory. ACPICA BZ 967, 912.

Updates for Unix makefiles:
1) Add -D_FORTIFY_SOURCE=2 for gcc generation. Arjan van de Ven.
2) Update linker flags (move to end of command line) for AcpiExec utility.
Guan Chao.

Split ACPICA initialization functions to new file, utxfinit.c. Split from
utxface.c to improve modularity and reduce file size.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

  Previous Release:
    Non-Debug Version: 93.5K Code, 25.3K Data, 118.8K Total

    Debug Version: 173.7K Code, 74.0K Data, 247.7K Total
  Current Release:
    Non-Debug Version: 93.8K Code, 25.3K Data, 119.1K Total
    Debug Version: 175.7K Code, 74.8K Data, 250.5K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a problem with constant folding for fixed-length constant
expressions. The constant-folding code was not being invoked for constant
expressions that allow the use of type 3/4/5 opcodes to generate constants
for expressions such as ByteConstExpr, WordConstExpr, etc. This could result
in the generation of invalid AML bytecode. ACPICA BZ 970.

iASL: Fixed a generation issue on newer versions of Bison. Newer versions
apparently automatically emit some of the necessary externals. This change
handles these versions in order to eliminate generation warnings.

Disassembler: Added support to decode the DBG2 and CSRT ACPI tables.

Disassembler: Add support to decode _PLD buffers. The decoded buffer appears
within comments in the output file.

Debugger: Fixed a regression with the "Threads" command where
AE_BAD_PARAMETER was always returned.

----
11 July 2012. Summary of changes for version 20120711:

1) ACPICA Kernel-resident Subsystem:

Fixed a possible fault in the return package object repair code. Fixes a
problem that can occur when a lone package object is wrapped with an outer
package object in order to force conformance to the ACPI specification. Can
affect these predefined names: _ALR, _MLS, _PSS, _TRT, _TSS, _PRT, _HPX,
_DLM,
_CSD, _PSD, _TSD.

Removed code to disable/enable bus master arbitration (ARB_DIS bit in the
PM2_CNT register) in the ACPICA sleep/wake interfaces. Management of the
ARB_DIS bit must be implemented in the host-dependent C3 processor power
state
support. Note, ARB_DIS is obsolete and only applies to older chipsets, both
Intel and other vendors. (for Intel: ICH4-M and earlier)

This change removes the code to disable/enable bus master arbitration during
suspend/resume. Use of the ARB_DIS bit in the optional PM2_CNT register
causes
resume problems on some machines. The change has been in use for over seven
years within Linux.

Implemented two new external interfaces to support host-directed dynamic
ACPI
table load and unload. They are intended to simplify the host
implementation
of hot-plug support:
  AcpiLoadTable: Load an SSDT from a buffer into the namespace.

```
  AcpiUnloadParentTable: Unload an SSDT via a named object owned by the
table.
See the ACPICA reference for additional details. Adds one new file,
components/tables/tbxfload.c

Implemented and deployed two new interfaces for errors and warnings that
are
known to be caused by BIOS/firmware issues:
  AcpiBiosError: Prints "ACPI Firmware Error" message.
  AcpiBiosWarning: Prints "ACPI Firmware Warning" message.
Deployed these new interfaces in the ACPICA Table Manager code for ACPI
table
and FADT errors. Additional deployment to be completed as appropriate in
the
future. The associated conditional macros are ACPI_BIOS_ERROR and
ACPI_BIOS_WARNING. See the ACPICA reference for additional details.
ACPICA
BZ
843.

Implicit notify support: ensure that no memory allocation occurs within a
critical region. This fix moves a memory allocation outside of the time
that a
spinlock is held. Fixes issues on systems that do not allow this
behavior.
Jung-uk Kim.

Split exception code utilities and tables into a new file,
utilities/utexcep.c

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug
version of the code includes the debug output trace mechanism and has a
much
larger code and data size.

  Previous Release:
    Non-Debug Version:  93.1K Code, 25.1K Data, 118.2K Total
    Debug Version:     172.9K Code, 73.6K Data, 246.5K Total
  Current Release:
    Non-Debug Version:  93.5K Code, 25.3K Data, 118.8K Total
    Debug Version:     173.7K Code, 74.0K Data, 247.7K Total


2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a parser problem for hosts where EOF is defined as -1 instead
of
0. Jung-uk Kim.

Debugger: Enhanced the "tables" command to emit additional information
about
the current set of ACPI tables, including the owner ID and flags decode.

Debugger: Reimplemented the "unload" command to use the new
AcpiUnloadParentTable external interface. This command was disable
previously
due to need for an unload interface.

AcpiHelp: Added a new option to decode ACPICA exception codes. The -e
option
will decode 16-bit hex status codes (ACPI_STATUS) to name strings.

----------------------------------------
20 June 2012. Summary of changes for version 20120620:
```



```


1) ACPICA Kernel-resident Subsystem:

Implemented support to expand the "implicit notify" feature to allow
multiple
devices to be notified by a single GPE. This feature automatically
generates a
runtime device notification in the absence of a BIOS-provided GPE control
method (_Lxx/_Exx) or a host-installed handler for the GPE. Implicit
notify is
provided by ACPICA for Windows compatibility, and is a workaround for
BIOS
AML
code errors. See the description of the AcpiSetupGpeForWake interface in
the
APCICA reference. Bob Moore, Rafael Wysocki. ACPICA BZ 918.

Changed some comments and internal function names to simplify and ensure
correctness of the Linux code translation. No functional changes.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug
version of the code includes the debug output trace mechanism and has a
much
larger code and data size.

  Previous Release:
    Non-Debug Version:  93.0K Code, 25.1K Data, 118.1K Total
    Debug Version:     172.7K Code, 73.6K Data, 246.3K Total
  Current Release:
    Non-Debug Version:  93.1K Code, 25.1K Data, 118.2K Total
    Debug Version:     172.9K Code, 73.6K Data, 246.5K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler: Added support to emit short, commented descriptions for the
ACPI
predefined names in order to improve the readability of the disassembled
output. ACPICA BZ 959. Changes include:
  1) Emit descriptions for all standard predefined names (_INI, _STA,
_PRW,
etc.)
  2) Emit generic descriptions for the special names (_Exx, _Qxx, etc.)
  3) Emit descriptions for the resource descriptor names (_MIN, _LEN,
etc.)

AcpiSrc: Fixed several long-standing Linux code translation issues.
Argument
descriptions in function headers are now translated properly to lower
case
and
underscores. ACPICA BZ 961. Also fixes translation problems such as
these:
(old -> new)
  i_aSL -> iASL
  00-7_f -> 00-7F
  16_k -> 16K
  local_fADT -> local_FADT
  execute_oSI -> execute_OSI

iASL: Fixed a problem where null bytes were inadvertently emitted into
some
listing files.
```

iASL: Added the existing debug options to the standard help screen. There are
no longer two different help screens. ACPICA BZ 957.

AcpiHelp: Fixed some typos in the various predefined name descriptions. Also
expand some of the descriptions where appropriate.

iASL: Fixed the -ot option (display compile times/statistics). Was not working
properly for standard output; only worked for the debug file case.

---
18 May 2012. Summary of changes for version 20120518:

1) ACPICA Core Subsystem:

Added a new OSL interface, AcpiOsWaitEventsComplete. This interface is defined
to block until asynchronous events such as notifies and GPEs have completed.
Within ACPICA, it is only called before a notify or GPE handler is
removed/uninstalled. It also may be useful for the host OS within related
drivers such as the Embedded Controller driver. See the ACPICA reference for
additional information. ACPICA BZ 868.

ACPI Tables: Added a new error message for a possible overflow failure during
the conversion of FADT 32-bit legacy register addresses to internal common
64-
bit GAS structure representation. The GAS has a one-byte "bit length" field,
thus limiting the register length to 255 bits. ACPICA BZ 953.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug
version of the code includes the debug output trace mechanism and has a much
larger code and data size.

```
  Previous Release:
    Non-Debug Version:  92.9K Code, 25.0K Data, 117.9K Total
    Debug Version:     172.6K Code, 73.4K Data, 246.0K Total
  Current Release:
    Non-Debug Version:  93.0K Code, 25.1K Data, 118.1K Total
    Debug Version:     172.7K Code, 73.6K Data, 246.3K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Added the ACPI 5.0 "PCC" keyword for use in the Register() ASL macro.
This keyword was added late in the ACPI 5.0 release cycle and was not
implemented until now.

Disassembler: Added support for Operation Region externals. Adds missing
support for operation regions that are defined in another table, and
referenced locally via a Field or BankField ASL operator. Now generates the
correct External statement.

Disassembler: Several additional fixes for the External() statement generation
related to some ASL operators. Also, order the External() statements
alphabetically in the disassembler output. Fixes the External() generation for
the Create* field, Alias, and Scope operators:
 1) Create* buffer field operators - fix type mismatch warning on disassembly
 2) Alias - implement missing External support
 3) Scope - fix to make sure all necessary externals are emitted.

iASL: Improved pathname support. For include files, merge the prefix pathname
with the file pathname and eliminate unnecessary components. Convert
backslashes in all pathnames to forward slashes, for readability. Include file
pathname changes affect both #include and Include() type operators.

iASL/DTC/Preprocessor: Gracefully handle early EOF. Handle an EOF at the end
of a valid line by inserting a newline and then returning the EOF during the
next call to GetNextLine. Prevents the line from being ignored due to EOF
condition.

iASL: Implemented some changes to enhance the IDE support (-vi option.) Error
and Warning messages are now correctly recognized for both the source code
browser and the global error and warning counts.

---
20 April 2012. Summary of changes for version 20120420:

1) ACPICA Core Subsystem:

Implemented support for multiple notify handlers. This change adds support to
allow multiple system and device notify handlers on Device, Thermal Zone, and
Processor objects. This can simplify the host OS notification implementation.
Also re-worked and restructured the entire notify support code to simplify
handler installation, handler removal, notify event queuing, and notify
dispatch to handler(s). Note: there can still only be two global notify
handlers - one for system notifies and one for device notifies. There are no
changes to the existing handler install/remove interfaces. Lin Ming, Bob
Moore, Rafael Wysocki.

Fixed a regression in the package repair code where the object reference
count was calculated incorrectly. Regression was introduced in the commit
"Support to add Package wrappers".

Fixed a couple possible memory leaks in the AML parser, in the error recovery
path. Jesper Juhl, Lin Ming.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a

much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  92.9K Code, 25.0K Data, 117.9K Total
    Debug Version:     172.5K Code, 73.2K Data, 245.7K Total
  Current Release:
    Non-Debug Version:  92.9K Code, 25.0K Data, 117.9K Total
    Debug Version:     172.6K Code, 73.4K Data, 246.0K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a problem with the resource descriptor support where the length
of the StartDependentFn and StartDependentFnNoPrio descriptors were not
included in cumulative descriptor offset, resulting in incorrect values for
resource tags within resource descriptors appearing after a StartDependent*
descriptor. Reported by Petr Vandrovec. ACPICA BZ 949.

iASL and Preprocessor: Implemented full support for the #line directive to
correctly track original source file line numbers through the .i preprocessor
output file - for error and warning messages.

iASL: Expand the allowable byte constants for address space IDs. Previously,
the allowable range was 0x80-0xFF (user-defined spaces), now the range is
0x0A-0xFF to allow for custom and new IDs without changing the compiler.

iASL: Add option to treat all warnings as errors (-we). ACPICA BZ 948.

iASL: Add option to completely disable the preprocessor (-Pn).

iASL: Now emit all error/warning messages to standard error (stderr) by
default (instead of the previous stdout).

ASL Test Suite (ASLTS): Reduce iASL warnings due to use of Switch(). Update
for resource descriptor offset fix above. Update/cleanup error output
routines. Enable and send iASL errors/warnings to an error logfile
(error.txt). Send all other iASL output to a logfile (compiler.txt). Fixed
several extraneous "unrecognized operator" messages.

---

20 March 2012. Summary of changes for version 20120320:

1) ACPICA Core Subsystem:

Enhanced the sleep/wake interfaces to optionally execute the _GTS method
(Going To Sleep) and the _BFS method (Back From Sleep). Windows apparently
does not execute these methods, and therefore these methods are often
untested. It has been seen on some systems where the execution of these
methods causes errors and also prevents the machine from entering S5. It is
therefore suggested that host operating systems do not execute these methods
by default. In the future, perhaps these methods can be optionally executed
based on the age of the system and/or what is the newest version of Windows

that the BIOS asks for via _OSI. Changed interfaces: AcpiEnterSleepState and
AcpileaveSleepStatePrep. See the ACPICA reference and Linux BZ 13041. Lin
Ming.

Fixed a problem where the length of the local/common FADT was set too early.
The local FADT table length cannot be set to the common length until the
original length has been examined. There is code that checks the table length
and sets various fields appropriately. This can affect older machines with
early FADT versions. For example, this can cause inadvertent writes to the
CST_CNT register. Julian Anastasov.

Fixed a mapping issue related to a physical table override. Use the deferred
mapping mechanism for tables loaded via the physical override OSL interface.
This allows for early mapping before the virtual memory manager is available.
Thomas Renninger, Bob Moore.

Enhanced the automatic return-object repair code: Repair a common problem with
predefined methods that are defined to return a variable-length Package of
sub-objects. If there is only one sub-object, some BIOS ASL code mistakenly
simply returns the single object instead of a Package with one sub-object.
This new support will repair this error by wrapping a Package object around
the original object, creating the correct and expected Package with one sub-
object. Names that can be repaired in this manner include: _ALR, _CSD, _HPX,
_MLS, _PLD, _PRT, _PSS, _TRT, _TSS, _BCL, _DOD, _FIX, and _Sx. ACPICA BZ 939.

Changed the exception code returned for invalid ACPI paths passed as
parameters to external interfaces such as AcpiEvaluateObject. Was
AE_BAD_PARAMETER, now is the more sensible AE_BAD_PATHNAME.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug
version of the code includes the debug output trace mechanism and has a much
larger code and data size.

```
  Previous Release:
    Non-Debug Version:  93.0K Code, 25.0K Data, 118.0K Total
    Debug Version:     172.5K Code, 73.2K Data, 245.7K Total
  Current Release:
    Non-Debug Version:  92.9K Code, 25.0K Data, 117.9K Total
    Debug Version:     172.5K Code, 73.2K Data, 245.7K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Added the infrastructure and initial implementation of a integrated C-
like preprocessor. This will simplify BIOS development process by eliminating

the need for a separate preprocessing step during builds. On Windows, it also
eliminates the need to install a separate C compiler. ACPICA BZ 761. Some
features including full #define() macro support are still under development.
These preprocessor directives are supported:
    #define
    #elif
    #else
    #endif
    #error
    #if
    #ifdef
    #ifndef
    #include
    #pragma message
    #undef
    #warning
In addition, these new command line options are supported:
    -D <symbol> Define symbol for preprocessor use
    -li   Create preprocessed output file (*.i)
    -P   Preprocess only and create preprocessor output file (*.i)

Table Compiler: Fixed a problem where the equals operator within an expression
did not work properly.

Updated iASL to use the current versions of Bison/Flex. Updated the Windows
project file to invoke these tools from the standard location. ACPICA BZ 904.
Versions supported:
    Flex for Windows: V2.5.4
    Bison for Windows: V2.4.1

----
15 February 2012. Summary of changes for version 20120215:

1) ACPICA Core Subsystem:

There have been some major changes to the sleep/wake support code, as
described below (a - e).

a) The AcpiLeaveSleepState has been split into two interfaces, similar to
AcpiEnterSleepStatePrep and AcpiEnterSleepState. The new interface is
AcpiLeaveSleepStatePrep. This allows the host to perform actions between the
time the _BFS method is called and the _WAK method is called. NOTE: all hosts
must update their wake/resume code or else sleep/wake will not work properly.
Rafael Wysocki.

b) In AcpiLeaveSleepState, now enable all runtime GPEs before calling the _WAK
method. Some machines require that the GPEs are enabled before the _WAK method
is executed. Thomas Renninger.

c) In AcpiLeaveSleepState, now always clear the WAK_STS (wake status) bit.
Some BIOS code assumes that WAK_STS will be cleared on resume and use it to
determine whether the system is rebooting or resuming. Matthew Garrett.

d) Move the invocations of _GTS (Going To Sleep) and _BFS (Back From Sleep) to
match the ACPI specification requirement. Rafael Wysocki.

e) Implemented full support for the ACPI 5.0 SleepStatus and SleepControl
registers within the V5 FADT. This support adds two new files:
hardware/hwesleep.c implements the support for the new registers. Moved all
sleep/wake external interfaces to hardware/hwxfsleep.c.

Added a new OSL interface for ACPI table overrides,
AcpiOsPhysicalTableOverride. This interface allows the host to override a
table via a physical address, instead of the logical address required by
AcpiOsTableOverride. This simplifies the host implementation. Initial
implementation by Thomas Renninger. The ACPICA implementation creates a single
shared function for table overrides that attempts both a logical and a
physical override.

Expanded the OSL memory read/write interfaces to 64-bit data
(AcpiOsReadMemory, AcpiOsWriteMemory.) This enables full 64-bit memory
transfer support for GAS register structures passed to AcpiRead and
AcpiWrite.

Implemented the ACPI_REDUCED_HARDWARE option to allow the creation of a custom
build of ACPICA that supports only the ACPI 5.0 reduced hardware (SoC) model.
See the ACPICA reference for details. ACPICA BZ 942. This option removes about
10% of the code and 5% of the static data, and the following hardware ACPI
features become unavailable:
    PM Event and Control registers
    SCI interrupt (and handler)
    Fixed Events
    General Purpose Events (GPEs)
    Global Lock
    ACPI PM timer
    FACS table (Waking vectors and Global Lock)

Updated the unix tarball directory structure to match the ACPICA git source
tree. This ensures that the generic unix makefiles work properly (in
generate/unix). Also updated the Linux makefiles to match. ACPICA BZ 867.

Updated the return value of the _REV predefined method to integer value 5 to
reflect ACPI 5.0 support.

Moved the external ACPI PM timer interface prototypes to the public acpixf.h
file where they belong.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug
version of the code includes the debug output trace mechanism and has a much
larger code and data size.

  Previous Release:
    Non-Debug Version: 92.8K Code, 24.9K Data, 117.7K Total
    Debug Version: 171.7K Code, 72.9K Data, 244.5K Total

```
  Current Release:
    Non-Debug Version:  93.0K Code, 25.0K Data, 118.0K Total
    Debug Version:     172.5K Code, 73.2K Data, 245.7K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler: Fixed a problem with the new ACPI 5.0 serial resource
descriptors (I2C, SPI, UART) where the resource produce/consumer bit was
incorrectly displayed.

AcpiHelp: Add display of ACPI/PNP device IDs that are defined in the ACPI
specification.

----------------------------------------
11 January 2012. Summary of changes for version 20120111:


1) ACPICA Core Subsystem:

Implemented a new mechanism to allow host device drivers to check for
address
range conflicts with ACPI Operation Regions. Both SystemMemory and
SystemIO
address spaces are supported. A new external interface,
AcpiCheckAddressRange,
allows drivers to check an address range against the ACPI namespace. See
the
ACPICA reference for additional details. Adds one new file,
utilities/utaddress.c. Lin Ming, Bob Moore.

Fixed several issues with the ACPI 5.0 FADT support: Add the sleep
Control
and
Status registers, update the ACPI 5.0 flags, and update internal data
structures to handle an FADT larger than 256 bytes. The size of the ACPI
5.0
FADT is 268 bytes.

Updated all ACPICA copyrights and signons to 2012. Added the 2012
copyright to
all module headers and signons, including the standard Linux header. This
affects virtually every file in the ACPICA core subsystem, iASL compiler,
and
all ACPICA utilities.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug
version of the code includes the debug output trace mechanism and has a
much
larger code and data size.

  Previous Release:
    Non-Debug Version:  92.3K Code, 24.9K Data, 117.2K Total
    Debug Version:     170.8K Code, 72.6K Data, 243.4K Total
  Current Release:
    Non-Debug Version:  92.8K Code, 24.9K Data, 117.7K Total
    Debug Version:     171.7K Code, 72.9K Data, 244.5K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler: fixed a problem with the automatic resource tag generation
support. Fixes a problem where the resource tags are inadvertently not
constructed if the table being disassembled contains external references
```



```
to
control methods. Moved the actual construction of the tags to after the
final
namespace is constructed (after 2nd parse is invoked due to external
control
method references.) ACPICA BZ 941.

Table Compiler: Make all "generic" operators caseless. These are the
operators
like UINT8, String, etc. Making these caseless improves ease-of-use.
ACPICA BZ
934.

----------------------------------------
23 November 2011. Summary of changes for version 20111123:

0) ACPI 5.0 Support:

This release contains full support for the ACPI 5.0 specification, as
summarized below.

Reduced Hardware Support:
-------------------------

This support allows for ACPI systems without the usual ACPI hardware.
This
support is enabled by a flag in the revision 5 FADT. If it is set, ACPICA
will
not attempt to initialize or use any of the usual ACPI hardware. Note,
when
this flag is set, all of the following ACPI hardware is assumed to be not
present and is not initialized or accessed:

    General Purpose Events (GPEs)
    Fixed Events (PM1a/PM1b and PM Control)
    Power Management Timer and Console Buttons (power/sleep)
    Real-time Clock Alarm
    Global Lock
    System Control Interrupt (SCI)
    The FACS is assumed to be non-existent

ACPI Tables:
------------

All new tables and updates to existing tables are fully supported in the
ACPICA headers (for use by device drivers), the disassembler, and the
iASL
Data Table Compiler. ACPI 5.0 defines these new tables:

    BGRT        /* Boot Graphics Resource Table */
    DRTM        /* Dynamic Root of Trust for Measurement table */
    FPDT        /* Firmware Performance Data Table */
    GTDT        /* Generic Timer Description Table */
    MPST        /* Memory Power State Table */
    PCCT        /* Platform Communications Channel Table */
    PMTT        /* Platform Memory Topology Table */
    RASF        /* RAS Feature table */

Operation Regions/SpaceIDs:
---------------------------

All new operation regions are fully supported by the iASL compiler, the
disassembler, and the ACPICA runtime code (for dispatch to region
handlers.)
The new operation region Space IDs are:
```

```
    GeneralPurposeIo
    GenericSerialBus

Resource Descriptors:
---------------------

All new ASL resource descriptors are fully supported by the iASL
compiler,
the
ASL/AML disassembler, and the ACPICA runtime Resource Manager code
(including
all new predefined resource tags). New descriptors are:

    FixedDma
    GpioIo
    GpioInt
    I2cSerialBus
    SpiSerialBus
    UartSerialBus

ASL/AML Operators, New and Modified:
------------------------------------

One new operator is added, the Connection operator, which is used to
associate
a GeneralPurposeIo or GenericSerialBus resource descriptor with
individual
field objects within an operation region. Several new protocols are
associated
with the AccessAs operator. All are fully supported by the iASL compiler,
disassembler, and runtime ACPICA AML interpreter:

    Connection                      // Declare Field Connection
attributes
    AccessAs: AttribBytes (n)           // Read/Write N-Bytes Protocol
    AccessAs: AttribRawBytes (n)        // Raw Read/Write N-Bytes
Protocol
    AccessAs: AttribRawProcessBytes (n) // Raw Process Call Protocol
    RawDataBuffer                       // Data type for Vendor Data
fields

Predefined ASL/AML Objects:
---------------------------

All new predefined objects/control-methods are supported by the iASL
compiler
and the ACPICA runtime validation/repair (arguments and return values.)
New
predefined names include the following:

Standard Predefined Names (Objects or Control Methods):
    _AEI, _CLS, _CPC, _CWS, _DEP,
    _DLM, _EVT, _GCP, _CRT, _GWS,
    _HRV, _PRE, _PSE, _SRT, _SUB.

Resource Tags (Names used to access individual fields within resource
descriptors):
    _DBT, _DPL, _DRS, _END, _FLC,
    _IOR, _LIN, _MOD, _PAR, _PHA,
    _PIN, _PPI, _POL, _RXL, _SLV,
    _SPE, _STB, _TXL, _VEN.

ACPICA External Interfaces:
---------------------------

Several new interfaces have been defined for use by ACPI-related device
```



```
drivers and other host OS services:

AcpiAcquireMutex and AcpiReleaseMutex: These interfaces allow the host OS
to
acquire and release AML mutexes that are defined in the DSDT/SSDT tables
provided by the BIOS. They are intended to be used in conjunction with
the
ACPI 5.0 _DLM (Device Lock Method) in order to provide transaction-level
mutual exclusion with the AML code/interpreter.

AcpiGetEventResources: Returns the (formatted) resource descriptors as
defined
by the ACPI 5.0 _AEI object (ACPI Event Information).  This object
provides
resource descriptors associated with hardware-reduced platform events,
similar
to the AcpiGetCurrentResources interface.

Operation Region Handlers: For General Purpose IO and Generic Serial Bus
operation regions, information about the Connection() object and any
optional
length information is passed to the region handler within the Context
parameter.

AcpiBufferToResource: This interface converts a raw AML buffer containing
a
resource template or resource descriptor to the ACPI_RESOURCE internal
format
suitable for use by device drivers. Can be used by an operation region
handler
to convert the Connection() buffer object into a ACPI_RESOURCE.

Miscellaneous/Tools/TestSuites:
-------------------------------

Support for extended _HID names (Four alpha characters instead of three).
Support for ACPI 5.0 features in the AcpiExec and AcpiHelp utilities.
Support for ACPI 5.0 features in the ASLTS test suite.
Fully updated documentation (ACPICA and iASL reference documents.)

ACPI Table Definition Language:
-------------------------------

Support for this language was implemented and released as a subsystem of
the
iASL compiler in 2010. (See the iASL compiler User Guide.)


Non-ACPI 5.0 changes for this release:
--------------------------------------

1) ACPICA Core Subsystem:

Fix a problem with operation region declarations where a failure can
occur
if
the region name and an argument that evaluates to an object (such as the
region address) are in different namespace scopes. Lin Ming, ACPICA BZ
937.

Do not abort an ACPI table load if an invalid space ID is found within.
This
will be caught later if the offending method is executed. ACPICA BZ 925.

Fixed an issue with the FFixedHW space ID where the ID was not always
recognized properly (Both ACPICA and iASL). ACPICA BZ 926.
```

Fixed a problem with the 32-bit generation of the unix-specific OSL
(osunixxf.c). Lin Ming, ACPICA BZ 936.

Several changes made to enable generation with the GCC 4.6 compiler.
ACPICA BZ
935.

New error messages: Unsupported I/O requests (not 8/16/32 bit), and
Index/Bank
field registers out-of-range.

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented the __PATH__ operator, which returns the full pathname
of
the current source file.

AcpiHelp: Automatically display expanded keyword information for all ASL
operators.

Debugger: Add "Template" command to disassemble/dump resource template
buffers.

Added a new master script to generate and execute the ASLTS test suite.
Automatically handles 32- and 64-bit generation. See tests/aslts.sh

iASL: Fix problem with listing generation during processing of the
Switch()
operator where AML listing was disabled until the entire Switch block was
completed.

iASL: Improve support for semicolon statement terminators. Fix "invalid
character" message for some cases when the semicolon is used. Semicolons
are
now allowed after every <Term> grammar element. ACPICA BZ 927.

iASL: Fixed some possible aliasing warnings during generation. ACPICA BZ
923.

Disassembler: Fix problem with disassembly of the DataTableRegion
operator
where an inadvertent "Unhandled deferred opcode" message could be
generated.

3) Example Code and Data Size

These are the sizes for the OS-independent acpica.lib produced by the
Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code
includes the debug output trace mechanism and has a much larger code and
data
size.

```
  Previous Release:
    Non-Debug Version:  90.2K Code, 23.9K Data, 114.1K Total
    Debug Version:     165.6K Code, 68.4K Data, 234.0K Total
  Current Release:
    Non-Debug Version:  92.3K Code, 24.9K Data, 117.2K Total
    Debug Version:     170.8K Code, 72.6K Data, 243.4K Total
```

---

22 September 2011. Summary of changes for version 20110922:

0) ACPI 5.0 News:

Support for ACPI 5.0 in ACPICA has been underway for several months and

will
be released at the same time that ACPI 5.0 is officially released.

The ACPI 5.0 specification is on track for release in the next few
months.

1) ACPICA Core Subsystem:

Fixed a problem where the maximum sleep time for the Sleep() operator was
intended to be limited to two seconds, but was inadvertently limited to
20
seconds instead.

Linux and Unix makefiles: Added header file dependencies to ensure
correct
generation of ACPICA core code and utilities. Also simplified the
makefiles
considerably through the use of the vpath variable to specify search
paths.
ACPICA BZ 924.

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented support to check the access length for all fields
created to
access named Resource Descriptor fields. For example, if a resource field
is
defined to be two bits, a warning is issued if a CreateXxxxField() is
used
with an incorrect bit length. This is implemented for all current
resource
descriptor names. ACPICA BZ 930.

Disassembler: Fixed a byte ordering problem with the output of 24-bit and
56-
bit integers.

iASL: Fixed a couple of issues associated with variable-length package
objects. 1) properly handle constants like One, Ones, Zero -- do not make
a
VAR_PACKAGE when these are used as a package length. 2) Allow the
VAR_PACKAGE
opcode (in addition to PACKAGE) when validating object types for
predefined
names.

iASL: Emit statistics for all output files (instead of just the ASL input
and
AML output). Includes listings, hex files, etc.

iASL: Added -G option to the table compiler to allow the compilation of
custom
ACPI tables. The only part of a table that is required is the standard
36-
byte
ACPI header.

AcpiXtract: Ported to the standard ACPICA environment (with ACPICA
headers),
which also adds correct 64-bit support. Also, now all output filenames
are
completely lower case.

AcpiExec: Ignore any non-AML tables (tables other than DSDT or SSDT) when
loading table files. A warning is issued for any such tables. The only
exception is an FADT. This also fixes a possible fault when attempting to

load
non-AML tables. ACPICA BZ 932.

AcpiHelp: Added the AccessAs and Offset operators. Fixed a problem where
a
missing table terminator could cause a fault when using the -p option.

AcpiSrc: Fixed a possible divide-by-zero fault when generating file
statistics.

3) Example Code and Data Size

These are the sizes for the OS-independent acpica.lib produced by the
Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code
includes the debug output trace mechanism and has a much larger code and
data
size.

```
  Previous Release (VC 9.0):
    Non-Debug Version:  90.2K Code, 23.9K Data, 114.1K Total
    Debug Version:     165.6K Code, 68.4K Data, 234.0K Total
  Current Release (VC 9.0):
    Non-Debug Version:  90.2K Code, 23.9K Data, 114.1K Total
    Debug Version:     165.6K Code, 68.4K Data, 234.0K Total
```

---

23 June 2011. Summary of changes for version 20110623:

1) ACPI CA Core Subsystem:

Updated the predefined name repair mechanism to not attempt repair of a
_TSS
return object if a _PSS object is present. We can only sort the _TSS
return
package if there is no _PSS within the same scope. This is because if
_PSS
is
present, the ACPI specification dictates that the _TSS Power Dissipation
field
is to be ignored, and therefore some BIOSs leave garbage values in the
_TSS
Power field(s). In this case, it is best to just return the _TSS package
as-
is. Reported by, and fixed with assistance from Fenghua Yu.

Added an option to globally disable the control method return value
validation
and repair. This runtime option can be used to disable return value
repair
if
this is causing a problem on a particular machine. Also added an option
to
AcpiExec (-dr) to set this disable flag.

All makefiles and project files: Major changes to improve generation of
ACPICA
tools. ACPICA BZ 912:

- Reduce default optimization levels to improve compatibility
- For Linux, add strict-aliasing=0 for gcc 4
- Cleanup and simplify use of command line defines
- Cleanup multithread library support
- Improve usage messages

Linux-specific header: update handling of THREAD_ID and pthread. For the
32-

bit case, improve casting to eliminate possible warnings, especially with
the
acpica tools.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
debug
version of the code includes the debug output trace mechanism and has a
much
larger code and data size.

```
  Previous Release (VC 9.0):
    Non-Debug Version:  90.1K Code, 23.9K Data, 114.0K Total
    Debug Version:     165.6K Code, 68.4K Data, 234.0K Total
  Current Release (VC 9.0):
    Non-Debug Version:  90.2K Code, 23.9K Data, 114.1K Total
    Debug Version:     165.6K Code, 68.4K Data, 234.0K Total
```

2) iASL Compiler/Disassembler and Tools:

With this release, a new utility named "acpihelp" has been added to the
ACPICA
package. This utility summarizes the ACPI specification chapters for the
ASL
and AML languages. It generates under Linux/Unix as well as Windows, and
provides the following functionality:

- Find/display ASL operator(s) -- with description and syntax.
- Find/display ASL keyword(s) -- with exact spelling and descriptions.
- Find/display ACPI predefined name(s) -- with description, number of arguments, and the return value data type.
- Find/display AML opcode name(s) -- with opcode, arguments, and grammar.
- Decode/display AML opcode -- with opcode name, arguments, and grammar.

Service Layers: Make multi-thread support configurable. Conditionally
compile
the multi-thread support so that threading libraries will not be linked
if
not
necessary. The only tool that requires multi-thread support is AcpiExec.

iASL: Update yyerrror/AslCompilerError for "const" errors. Newer versions
of
Bison appear to want the interface to yyerror to be a const char * (or at
least this is a problem when generating iASL on some systems.) ACPICA BZ
923
Pierre Lejeune.

Tools: Fix for systems where O_BINARY is not defined. Only used for
Windows
versions of the tools.

---

27 May 2011. Summary of changes for version 20110527:

1) ACPI CA Core Subsystem:

ASL Load() operator: Reinstate most restrictions on the incoming ACPI
table
8 *ASL Load() operator: Reinstate most restrictions on the incoming ACPI table*
signature. Now, only allow SSDT, OEMx, and a null signature. History:

1) Originally, we checked the table signature for "SSDT" or "PSDT". (PSDT is now obsolete.)
2) We added support for OEMx tables, signature "OEM" plus a fourth "don't care" character.

```
     3) Valid tables were encountered with a null signature, so we just
        gave up on validating the signature, (05/2008).
     4) We encountered non-AML tables such as the MADT, which caused
        interpreter errors and kernel faults. So now, we once again allow
        only SSDT, OEMx, and now, also a null signature. (05/2011).

Added the missing _TDL predefined name to the global name list in order to
enable validation. Affects both the core ACPICA code and the iASL compiler.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug
version of the code includes the debug output trace mechanism and has a much
larger code and data size.

   Previous Release (VC 9.0):
     Non-Debug Version:  90.0K Code, 23.8K Data, 113.8K Total
     Debug Version:     164.5K Code, 68.0K Data, 232.5K Total
   Current Release (VC 9.0):
     Non-Debug Version:  90.1K Code, 23.9K Data, 114.0K Total
     Debug Version:     165.6K Code, 68.4K Data, 234.0K Total

2) iASL Compiler/Disassembler and Tools:

Debugger/AcpiExec: Implemented support for "complex" method arguments on the
debugger command line. This adds support beyond simple integers -- including
Strings, Buffers, and Packages. Includes support for nested packages.
Increased the default command line buffer size to accommodate these
arguments.
See the ACPICA reference for details and syntax. ACPICA BZ 917.

Debugger/AcpiExec: Implemented support for "default" method arguments for the
Execute/Debug command. Now, the debugger will always invoke a control method
with the required number of arguments -- even if the command line specifies
none or insufficient arguments. It uses default integer values for any missing
arguments. Also fixes a bug where only six method arguments maximum were
supported instead of the required seven.

Debugger/AcpiExec: Add a maximum buffer length parameter to AcpiOsGetLine and
also return status in order to prevent buffer overruns. See the ACPICA
reference for details and syntax. ACPICA BZ 921

iASL: Cleaned up support for Berkeley yacc. A general cleanup of code and
makefiles to simplify support for the two different but similar parser
```



```
generators, bison and yacc.

Updated the generic unix makefile for gcc 4. The default gcc version is now
expected to be 4 or greater, since options specific to gcc 4 are used.

----------------------------------------
13 April 2011. Summary of changes for version 20110413:

1) ACPI CA Core Subsystem:

Implemented support to execute a so-called "orphan" _REG method under the
device. This change will force the execution of a _REG method underneath the
EC
device. This change will force the execution of a _REG method underneath the
EC
device even if there is no corresponding operation region of type
EmbeddedControl. Fixes a problem seen on some machines and apparently is
compatible with Windows behavior. ACPICA BZ 875.

Added more predefined methods that are eligible for automatic NULL package
element removal. This change adds another group of predefined names to the
list
of names that can be repaired by having NULL package elements dynamically
removed. This group are those methods that return a single variable-length
package containing simple data types such as integers, buffers, strings. This
includes: _ALx, _BCL, _CID,_ DOD, _EDL, _FIX, _PCL, _PLD, _PMD, _PRx, _PSL,
_Sx,
and _TZD. ACPICA BZ 914.

Split and segregated all internal global lock functions to a new file,
evglock.c.

Updated internal address SpaceID for DataTable regions. Moved this internal
space
id in preparation for ACPI 5.0 changes that will include some new space IDs.
This
change should not affect user/host code.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib
produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of
the code includes the debug output trace mechanism and has a much larger code
and
data size.
```

```
   Previous Release (VC 9.0):
     Non-Debug Version:  89.8K Code, 23.8K Data, 113.6K Total
     Debug Version:     164.2K Code, 67.9K Data, 232.1K Total
   Current Release (VC 9.0):
     Non-Debug Version:  90.0K Code, 23.8K Data, 113.8K Total
     Debug Version:     164.5K Code, 68.0K Data, 232.5K Total

2) iASL Compiler/Disassembler and Tools:

iASL/DTC: Major update for new grammar features. Allow generic data types in
custom ACPI tables. Field names are now optional. Any line can be split to
multiple lines using the continuation char (\). Large buffers now use line-
continuation character(s) and no colon on the continuation lines. See the
grammar
update in the iASL compiler reference. ACPI BZ 910,911. Lin Ming, Bob Moore.
 iASL/DTC: Major update for new grammar features. Allow generic data types in
 custom ACPI tables. Field names are now optional. Any line can be split to
 multiple lines using the continuation char (\). Large buffers now use line-
 update in the iASL compiler reference. ACPI BZ 910,911. Lin Ming, Bob Moore.

iASL: Mark ASL "Return()" and the simple "Return" as "Null" return
statements.
Since the parser stuffs a "zero" as the return value for these statements
(due
to
the underlying AML grammar), they were seen as "return with value" by the
iASL
semantic checking. They are now seen correctly as "null" return
statements.
 iASL: Mark ASL "Return()" and the simple "Return" as "Null" return statements.
 Since the parser stuffs a "zero" as the return value for these statements (due
 the underlying AML grammar), they were seen as "return with value" by the iASL
 semantic checking. They are now seen correctly as "null" return statements.

iASL: Check if a_REG declaration has a corresponding Operation Region.
Adds a
check for each _REG to ensure that there is in fact a corresponding
operation
 iASL: Check if a_REG declaration has a corresponding Operation Region. Adds a
 check for each _REG to ensure that there is in fact a corresponding operation
region declaration in the same scope. If not, the _REG method is not very
useful
since it probably won't be executed. ACPICA BZ 915.

iASL/DTC: Finish support for expression evaluation. Added a new
expression
 iASL/DTC: Finish support for expression evaluation. Added a new expression
parser
that implements c-style operator precedence and parenthesization. ACPICA
bugzilla
908.

Disassembler/DTC: Remove support for () and <> style comments in data
tables.
 Disassembler/DTC: Remove support for () and <> style comments in data tables.
Now
that DTC has full expression support, we don't want to have comment
strings
 that DTC has full expression support, we don't want to have comment strings
that
start with a parentheses or a less-than symbol. Now, only the standard /*
```



```
and
 start with a parentheses or a less-than symbol. Now, only the standard /* and
//
comments are supported, as well as the bracket [] comments.

AcpiXtract: Fix for RSDP and dynamic SSDT extraction. These tables have
"unusual"
headers in the acpidump file. Update the header validation to support
these
tables. Problem introduced in previous AcpiXtract version in the change
to
 headers in the acpidump file. Update the header validation to support these
 tables. Problem introduced in previous AcpiXtract version in the change to
support "wrong checksum" error messages emitted by acpidump utility.

iASL: Add a * option to generate all template files (as a synonym for
ALL)
as
 iASL: Add a * option to generate all template files (as a synonym for ALL) as
in
"iasl -T *" or "iasl -T ALL".

iASL/DTC: Do not abort compiler on fatal errors. We do not want to
completely
abort the compiler on "fatal" errors, simply should abort the current
compile.
 iASL/DTC: Do not abort compiler on fatal errors. We do not want to completely
 abort the compiler on "fatal" errors, simply should abort the current compile.
This allows multiple compiles with a single (possibly wildcard) compiler
invocation.

---------------------------------------
16 March 2011. Summary of changes for version 20110316:

1) ACPI CA Core Subsystem:

Fixed a problem caused by a _PRW method appearing at the namespace root
scope
during the setup of wake GPEs. A fault could occur if a _PRW directly
under
 Fixed a problem caused by a _PRW method appearing at the namespace root scope
 during the setup of wake GPEs. A fault could occur if a _PRW directly under
the
root object was passed to the AcpiSetupGpeForWake interface. Lin Ming.

Implemented support for "spurious" Global Lock interrupts. On some
systems, a
global lock interrupt can occur without the pending flag being set. Upon
a
GL
interrupt, we now ensure that a thread is actually waiting for the lock
before
 Implemented support for "spurious" Global Lock interrupts. On some systems, a
 global lock interrupt can occur without the pending flag being set. Upon a GL
 interrupt, we now ensure that a thread is actually waiting for the lock before
signaling GL availability. Rafael Wysocki, Bob Moore.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib
produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug
version of
the code includes the debug output trace mechanism and has a much larger
code
 produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of
 the code includes the debug output trace mechanism and has a much larger code
and
```

```
2711 data size.

2713   Previous Release (VC 9.0):
2714     Non-Debug Version:  89.7K Code, 23.7K Data, 113.4K Total
2715     Debug Version:     163.9K Code, 67.5K Data, 231.4K Total
2716   Current Release (VC 9.0):
2717     Non-Debug Version:  89.8K Code, 23.8K Data, 113.6K Total
2718     Debug Version:     164.2K Code, 67.9K Data, 232.1K Total

2720 2) iASL Compiler/Disassembler and Tools:

2722 Implemented full support for the "SLIC" ACPI table. Includes support in
2723 the
2724 header files, disassembler, table compiler, and template generator. Bob
2725 Moore,
 187 Implemented full support for the "SLIC" ACPI table. Includes support in the
 188 header files, disassembler, table compiler, and template generator. Bob Moore,
2726 Lin Ming.

2728 AcpiXtract: Correctly handle embedded comments and messages from
2729 AcpiDump.
2730 Apparently some or all versions of acpidump will occasionally emit a
2731 comment
 191 AcpiXtract: Correctly handle embedded comments and messages from AcpiDump.
 192 Apparently some or all versions of acpidump will occasionally emit a comment
2732 like
2733 "Wrong checksum", etc., into the dump file. This was causing problems for
2734 AcpiXtract. ACPICA BZ 905.

2736 iASL: Fix the Linux makefile by removing an inadvertent double file
2737 inclusion.
 197 iASL: Fix the Linux makefile by removing an inadvertent double file inclusion.
2738 ACPICA BZ 913.

2740 AcpiExec: Update installation of operation region handlers. Install one
2741 handler
2742 for a user-defined address space. This is used by the ASL test suite
2743 (ASLTS).
 202 for a user-defined address space. This is used by the ASL test suite (ASLTS).

2745 ----------------------------------------
2746 11 February 2011. Summary of changes for version 20110211:

2748 1) ACPI CA Core Subsystem:

2750 Added a mechanism to defer _REG methods for some early-installed
2751 handlers.
2752 Most user handlers should be installed before call to
2753 AcpiEnableSubsystem.
 209 Added a mechanism to defer _REG methods for some early-installed handlers.
 210 Most user handlers should be installed before call to AcpiEnableSubsystem.
2754 However, Event handlers and region handlers should be installed after
2755 AcpiInitializeObjects. Override handlers for the "default" regions should
2756 be
 212 AcpiInitializeObjects. Override handlers for the "default" regions should be
2757 installed early, however. This change executes all _REG methods for the
2758 default regions (Memory/IO/PCI/DataTable) simultaneously to prevent any
2759 chicken/egg issues between them. ACPICA BZ 848.

2761 Implemented an optimization for GPE detection. This optimization will
2762 simply
 217 Implemented an optimization for GPE detection. This optimization will simply
2763 ignore GPE registers that contain no enabled GPEs -- there is no need to
2764 read the register since this information is available internally. This
2765 becomes more important on machines with a large GPE space. ACPICA
2766 bugzilla
```



```
 220 becomes more important on machines with a large GPE space. ACPICA bugzilla
2767 884. Lin Ming. Suggestion from Joe Liu.

2769 Removed all use of the highly unreliable FADT revision field. The
2770 revision
2771 number in the FADT has been found to be completely unreliable and cannot
2772 be
2773 trusted. Only the actual table length can be used to infer the version.
2774 This
2775 change updates the ACPICA core and the disassembler so that both no
2776 longer
 223 Removed all use of the highly unreliable FADT revision field. The revision
 224 number in the FADT has been found to be completely unreliable and cannot be
 225 trusted. Only the actual table length can be used to infer the version. This
 226 change updates the ACPICA core and the disassembler so that both no longer
2777 even look at the FADT version and instead depend solely upon the FADT
2778 length.

2780 Fix an unresolved name issue for the no-debug and no-error-message source
2781 generation cases. The _AcpiModuleName was left undefined in these cases,
2782 but
 231 generation cases. The _AcpiModuleName was left undefined in these cases, but
2783 it is actually needed as a parameter to some interfaces. Define
2784 _AcpiModuleName as a null string in these cases. ACPICA Bugzilla 888.

2786 Split several large files (makefiles and project files updated)
2787   utglobal.c   -> utdecode.c
2788   dbcomds.c    -> dbmethod.c dbnames.c
2789   dsopcode.c   -> dsargs.c dscontrol.c
2790   dsload.c     -> dsload2.c
2791   aslanalyze.c -> aslbtypes.c aslwalks.c

2793 Example Code and Data Size: These are the sizes for the OS-independent
2794 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
2795 debug version of the code includes the debug output trace mechanism and
2796 has
 244 debug version of the code includes the debug output trace mechanism and has
2797 a much larger code and data size.

2799   Previous Release (VC 9.0):
2800     Non-Debug Version:  89.7K Code, 23.7K Data, 113.4K Total
2801     Debug Version:     163.9K Code, 67.5K Data, 231.4K Total
2802   Current Release (VC 9.0):
2803     Non-Debug Version:  89.7K Code, 23.7K Data, 113.4K Total
2804     Debug Version:     163.9K Code, 67.5K Data, 231.4K Total

2806 2) iASL Compiler/Disassembler and Tools:

2808 iASL: Implemented the predefined macros __LINE__, __FILE__, and __DATE__.
2809 These are useful C-style macros with the standard definitions. ACPICA
2810 bugzilla 898.

2812 iASL/DTC: Added support for integer expressions and labels. Support for
2813 full
2814 expressions for all integer fields in all ACPI tables. Support for labels
2815 in
 260 iASL/DTC: Added support for integer expressions and labels. Support for full
 261 expressions for all integer fields in all ACPI tables. Support for labels in
2816 "generic" portions of tables such as UEFI. See the iASL reference manual.

2818 Debugger: Added a command to display the status of global handlers. The
2819 "handlers" command will display op region, fixed event, and miscellaneous
2820 global handlers. installation status -- and for op regions, whether
2821 default
 266 global handlers. installation status -- and for op regions, whether default
2822 or user-installed handler will be used.
```

iASL: Warn if reserved method incorrectly returns a value. Many predefined names are defined such that they do not return a value. If implemented as a
iASL: Warn if reserved method incorrectly returns a value. Many predefined names are defined such that they do not return a value. If implemented as a method, issue a warning if such a name explicitly returns a value. ACPICA Bugzilla 855.

iASL: Added detection of GPE method name conflicts. Detects a conflict where there are two GPE methods of the form _Lxy and _Exy in the same scope. (For
iASL: Added detection of GPE method name conflicts. Detects a conflict where there are two GPE methods of the form _Lxy and _Exy in the same scope. (For example, _L1D and _E1D in the same scope.) ACPICA bugzilla 848.

iASL/DTC: Fixed a couple input scanner issues with comments and line **numbers. Comment remover could get confused and miss a comment ending. Fixed**
numbers. Comment remover could get confused and miss a comment ending. Fixed a problem with line counter maintenance.

iASL/DTC: Reduced the severity of some errors from fatal to error. There is
iASL/DTC: Reduced the severity of some errors from fatal to error. There is no need to abort on simple errors within a field definition.

Debugger: Simplified the output of the help command. All help output now in
Debugger: Simplified the output of the help command. All help output now in a single screen, instead of help subcommands. ACPICA Bugzilla 897.

---
12 January 2011. Summary of changes for version 20110112:

1) ACPI CA Core Subsystem:

Fixed a race condition between method execution and namespace walks that can
Fixed a race condition between method execution and namespace walks that can possibly cause a fault. The problem was apparently introduced in version **20100528 as a result of a performance optimization that reduces the number of**
20100528 as a result of a performance optimization that reduces the number of namespace walks upon method exit by using the delete_namespace_subtree **function instead of the delete_namespace_by_owner function used previously.**
function instead of the delete_namespace_by_owner function used previously. Bug is a missing namespace lock in the delete_namespace_subtree function. dana.myers@oracle.com

Fixed several issues and a possible fault with the automatic "serialized" **method support. History: This support changes a method to "serialized" on the**
method support. History: This support changes a method to "serialized" on the fly if the method generates an AE_ALREADY_EXISTS error, indicating the **possibility that it cannot handle reentrancy. This fix repairs a couple of**
possibility that it cannot handle reentrancy. This fix repairs a couple of issues seen in the field, especially on machines with many cores:

    1) Delete method children only upon the exit of the last thread,
       so as to not delete objects out from under other running threads

      (and possibly causing a fault.)
    2) Set the "serialized" bit for the method only upon the exit of the
       Last thread, so as to not cause deadlock when running threads
       attempt to exit.
    3) Cleanup the use of the AML "MethodFlags" and internal method flags
       so that there is no longer any confusion between the two.

    Lin Ming, Bob Moore. Reported by dana.myers@oracle.com.

Debugger: Now lock the namespace for duration of a namespace dump. Prevents
Debugger: Now lock the namespace for duration of a namespace dump. Prevents issues if the namespace is changing dynamically underneath the debugger. Especially affects temporary namespace nodes, since the debugger displays these also.

Updated the ordering of include files. The ACPICA headers should appear **before any compiler-specific headers (stdio.h, etc.) so that acenv.h can set any necessary compiler-specific defines, etc. Affects the ACPI-related tools**
before any compiler-specific headers (stdio.h, etc.) so that acenv.h can set any necessary compiler-specific defines, etc. Affects the ACPI-related tools and utilities.

Updated all ACPICA copyrights and signons to 2011. Added the 2011 copyright to all module headers and signons, including the Linux header. This affects
Updated all ACPICA copyrights and signons to 2011. Added the 2011 copyright to all module headers and signons, including the Linux header. This affects virtually every file in the ACPICA core subsystem, iASL compiler, and all utilities.

Added project files for MS Visual Studio 2008 (VC++ 9.0). The original **project files for VC++ 6.0 are now obsolete. New project files can be found**
project files for VC++ 6.0 are now obsolete. New project files can be found under acpica/generate/msvc9. See acpica/generate/msvc9/readme.txt for details.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

  Previous Release (VC 6.0):
    Non-Debug Version: 89.8K Code, 18.9K Data, 108.7K Total
    Debug Version: 166.6K Code, 52.1K Data, 218.7K Total
  Current Release (VC 9.0):
    Non-Debug Version: 89.7K Code, 23.7K Data, 113.4K Total
    Debug Version: 163.9K Code, 67.5K Data, 231.4K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Added generic data types to the Data Table compiler. Add "generic" data types such as UINT32, String, Unicode, etc., to simplify the generation of
iASL: Added generic data types to the Data Table compiler. Add "generic" data types such as UINT32, String, Unicode, etc., to simplify the generation of platform-defined tables such as UEFI. Lin Ming.

iASL: Added listing support for the Data Table Compiler. Adds listing

support
iASL: Added listing support for the Data Table Compiler. Adds listing support
(-l) to display actual binary output for each line of input code.

---
09 December 2010. Summary of changes for version 20101209:

1) ACPI CA Core Subsystem:

Completed the major overhaul of the GPE support code that was begun in
July
Completed the major overhaul of the GPE support code that was begun in July
2010. Major features include: removal of _PRW execution in ACPICA (host
executes _PRWs anyway), cleanup of "wake" GPE interfaces and processing,
changes to existing interfaces, simplification of GPE handler operation,
and
changes to existing interfaces, simplification of GPE handler operation, and
a handful of new interfaces:

    AcpiUpdateAllGpes
    AcpiFinishGpe
    AcpiSetupGpeForWake
    AcpiSetGpeWakeMask
    One new file, evxfgpe.c to consolidate all external GPE interfaces.

See the ACPICA Programmer Reference for full details and programming
information. See the new section 4.4 "General Purpose Event (GPE)
Support"
for a full overview, and section 8.7 "ACPI General Purpose Event
Management"
for programming details. ACPICA BZ 858,870,877. Matthew Garrett, Lin
Ming,
information. See the new section 4.4 "General Purpose Event (GPE) Support"
for a full overview, and section 8.7 "ACPI General Purpose Event Management"
for programming details. ACPICA BZ 858,870,877. Matthew Garrett, Lin Ming,
Bob Moore, Rafael Wysocki.

Implemented a new GPE feature for Windows compatibility, the "Implicit
Wake
GPE Notify". This feature will automatically issue a Notify(2) on a
device
Implemented a new GPE feature for Windows compatibility, the "Implicit Wake
GPE Notify". This feature will automatically issue a Notify(2) on a device
when a Wake GPE is received if there is no corresponding GPE method or
handler. ACPICA BZ 870.

Fixed a problem with the Scope() operator during table parse and load
phase.
During load phase (table load or method execution), the scope operator
should
not enter the target into the namespace. Instead, it should open a new
scope
Fixed a problem with the Scope() operator during table parse and load phase.
During load phase (table load or method execution), the scope operator should
not enter the target into the namespace. Instead, it should open a new scope
at the target location. Linux BZ 19462, ACPICA BZ 882.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

  Previous Release:
    Non-Debug Version: 89.8K Code, 18.9K Data, 108.7K Total

    Debug Version: 166.6K Code, 52.1K Data, 218.7K Total
  Current Release:
    Non-Debug Version: 89.9K Code, 19.0K Data, 108.9K Total
    Debug Version: 166.3K Code, 52.1K Data, 218.4K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Relax the alphanumeric restriction on _CID strings. These strings
are
"bus-specific" per the ACPI specification, and therefore any characters
are
acceptable. The only checks that can be performed are for a null string
and
iASL: Relax the alphanumeric restriction on _CID strings. These strings are
"bus-specific" per the ACPI specification, and therefore any characters are
acceptable. The only checks that can be performed are for a null string and
perhaps for a leading asterisk. ACPICA BZ 886.

iASL: Fixed a problem where a syntax error that caused a premature EOF
condition on the source file emitted a very confusing error message. The
premature EOF is now detected correctly. ACPICA BZ 891.

Disassembler: Decode the AccessSize within a Generic Address Structure
(byte
Disassembler: Decode the AccessSize within a Generic Address Structure (byte
access, word access, etc.) Note, this field does not allow arbitrary bit
access, the size is encoded as 1=byte, 2=word, 3=dword, and 4=qword.

New: AcpiNames utility - Example namespace dump utility. Shows an example
of
New: AcpiNames utility - Example namespace dump utility. Shows an example of
ACPICA configuration for a minimal namespace dump utility. Uses table and
namespace managers, but no AML interpreter. Does not add any
functionality
namespace managers, but no AML interpreter. Does not add any functionality
over AcpiExec, it is a subset of AcpiExec. The purpose is to show how to
partition and configure ACPICA. ACPICA BZ 883.

AML Debugger: Increased the debugger buffer size for method return
objects.
Was 4K, increased to 16K. Also enhanced error messages for debugger
method
AML Debugger: Increased the debugger buffer size for method return objects.
Was 4K, increased to 16K. Also enhanced error messages for debugger method
execution, including the buffer overflow case.

---
13 October 2010. Summary of changes for version 20101013:

1) ACPI CA Core Subsystem:

Added support to clear the PCIEXP_WAKE event. When clearing ACPI events,
now
Added support to clear the PCIEXP_WAKE event. When clearing ACPI events, now
clear the PCIEXP_WAKE_STS bit in the ACPI PM1 Status Register, via
HwClearAcpiStatus. Original change from Colin King. ACPICA BZ 880.

Changed the type of the predefined namespace object _TZ from ThermalZone
to
Device. This was found to be confusing to the host software that
processes
the various thermal zones, since _TZ is not really a ThermalZone.
However,
a
Changed the type of the predefined namespace object _TZ from ThermalZone to
Device. This was found to be confusing to the host software that processes

the various thermal zones, since _TZ is not really a ThermalZone. However, a
Notify() can still be performed on it. ACPICA BZ 876. Suggestion from Rui
Zhang.

Added Windows Vista SP2 to the list of supported _OSI strings. The actual
string is "Windows 2006 SP2".

Eliminated duplicate code in AcpiUtExecute* functions. Now that the
nsrepair
Eliminated duplicate code in AcpiUtExecute functions. Now that the nsrepair*
code automatically repairs _HID-related strings, this type of code is no
longer needed in Execute_HID, Execute_CID, and Execute_UID. ACPICA BZ
878.
longer needed in Execute_HID, Execute_CID, and Execute_UID. ACPICA BZ 878.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  89.9K Code, 19.0K Data, 108.9K Total
    Debug Version:     166.3K Code, 52.1K Data, 218.4K Total
  Current Release:
    Non-Debug Version:  89.9K Code, 19.0K Data, 108.9K Total
    Debug Version:     166.3K Code, 52.1K Data, 218.4K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented additional compile-time validation for _HID strings.
The
non-hex prefix (such as "PNP" or "ACPI") must be uppercase, and the
length
of
the string must be exactly seven or eight characters. For both _HID and
_CID
iASL: Implemented additional compile-time validation for _HID strings. The
non-hex prefix (such as "PNP" or "ACPI") must be uppercase, and the length of
the string must be exactly seven or eight characters. For both _HID and _CID
strings, all characters must be alphanumeric. ACPICA BZ 874.

iASL: Allow certain "null" resource descriptors. Some BIOS code creates
descriptors that are mostly or all zeros, with the expectation that they
will
be filled in at runtime. iASL now allows this as long as there is a
"resource
descriptors that are mostly or all zeros, with the expectation that they will
be filled in at runtime. iASL now allows this as long as there is a "resource
tag" (name) associated with the descriptor, which gives the ASL a handle
needed to modify the descriptor. ACPICA BZ 873.

Added single-thread support to the generic Unix application OSL.
Primarily
for iASL support, this change removes the use of semaphores in the
single-
Added single-thread support to the generic Unix application OSL. Primarily
for iASL support, this change removes the use of semaphores in the single-
threaded ACPICA tools/applications - increasing performance. The
_MULTI_THREADED option was replaced by the (reverse) ACPI_SINGLE_THREADED
option. ACPICA BZ 879.

AcpiExec: several fixes for the 64-bit version. Adds XSDT support and
support
AcpiExec: several fixes for the 64-bit version. Adds XSDT support and support

for 64-bit DSDT/FACS addresses in the FADT. Lin Ming.

iASL: Moved all compiler messages to a new file, aslmessages.h.

---
15 September 2010. Summary of changes for version 20100915:

1) ACPI CA Core Subsystem:

Removed the AcpiOsDerivePciId OSL interface. The various host
implementations
Removed the AcpiOsDerivePciId OSL interface. The various host implementations
of this function were not OS-dependent and are now obsolete and can be
removed from all host OSLs. This function has been replaced by
AcpiHwDerivePciId, which is now part of the ACPICA core code.
AcpiHwDerivePciId has been implemented without recursion. Adds one new
module, hwpci.c. ACPICA BZ 857.

Implemented a dynamic repair for _HID and _CID strings. The following
problems are now repaired at runtime: 1) Remove a leading asterisk in the
string, and 2) the entire string is uppercased. Both repairs are in
accordance with the ACPI specification and will simplify host driver
code.
accordance with the ACPI specification and will simplify host driver code.
ACPICA BZ 871.

The ACPI_THREAD_ID type is no longer configurable, internally it is now
always UINT64. This simplifies the ACPICA code, especially any printf
output.
always UINT64. This simplifies the ACPICA code, especially any printf output.
UINT64 is the only common data type for all thread_id types across all
operating systems. It is now up to the host OSL to cast the native
thread_id
type to UINT64 before returning the value to ACPICA (via
AcpiOsGetThreadId).
operating systems. It is now up to the host OSL to cast the native thread_id
type to UINT64 before returning the value to ACPICA (via AcpiOsGetThreadId).
Lin Ming, Bob Moore.

Added the ACPI_INLINE type to enhance the ACPICA configuration. The
"inline"
keyword is not standard across compilers, and this type allows inline to
be
Added the ACPI_INLINE type to enhance the ACPICA configuration. The "inline"
keyword is not standard across compilers, and this type allows inline to be
configured on a per-compiler basis. Lin Ming.

Made the system global AcpiGbl_SystemAwakeAndRunning publically
available.
Added an extern for this boolean in acpixf.h. Some hosts utilize this
value
Made the system global AcpiGbl_SystemAwakeAndRunning publically available.
Added an extern for this boolean in acpixf.h. Some hosts utilize this value
during suspend/restore operations. ACPICA BZ 869.

All code that implements error/warning messages with the "ACPI:" prefix
has
All code that implements error/warning messages with the "ACPI:" prefix has
been moved to a new module, utxferror.c.

The UINT64_OVERLAY was moved to utmath.c, which is the only module where
it
The UINT64_OVERLAY was moved to utmath.c, which is the only module where it
is used. ACPICA BZ 829. Lin Ming, Bob Moore.

Example Code and Data Size: These are the sizes for the OS-independent

acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
**debug version of the code includes the debug output trace mechanism and
has a**
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  89.1K Code, 19.0K Data, 108.1K Total
    Debug Version:     165.1K Code, 51.9K Data, 217.0K Total
  Current Release:
    Non-Debug Version:  89.9K Code, 19.0K Data, 108.9K Total
    Debug Version:     166.3K Code, 52.1K Data, 218.4K Total
```

2) iASL Compiler/Disassembler and Tools:

**iASL/Disassembler: Write ACPI errors to stderr instead of the output
file.
This keeps the output files free of random error messages that may
originate
from within the namespace/interpreter code. Used this opportunity to
merge**
*iASL/Disassembler: Write ACPI errors to stderr instead of the output file.
This keeps the output files free of random error messages that may originate
from within the namespace/interpreter code. Used this opportunity to merge*
all ACPI:-style messages into a single new module, utxferror.c. ACPICA BZ
866. Lin Ming, Bob Moore.

**Tools: update some printfs for ansi warnings on size_t. Handle width
change**
Tools: update some printfs for ansi warnings on size_t. Handle width change
of size_t on 32-bit versus 64-bit generations. Lin Ming.

---

06 August 2010. Summary of changes for version 20100806:

1) ACPI CA Core Subsystem:

**Designed and implemented a new host interface to the _OSI support code.
This
will allow the host to dynamically add or remove multiple _OSI strings,
as
well as install an optional handler that is called for each _OSI
invocation.
Also added a new AML debugger command, 'osi' to display and modify the
global
_OSI string table, and test support in the AcpiExec utility. See the
ACPICA**
*Designed and implemented a new host interface to the _OSI support code. This
will allow the host to dynamically add or remove multiple _OSI strings, as
well as install an optional handler that is called for each _OSI invocation.
Also added a new AML debugger command, 'osi' to display and modify the global
_OSI string table, and test support in the AcpiExec utility. See the ACPICA*
reference manual for full details. Lin Ming, Bob Moore. ACPICA BZ 836.
New Functions:
    AcpiInstallInterface - Add an _OSI string.
    AcpiRemoveInterface - Delete an _OSI string.
    AcpiInstallInterfaceHandler - Install optional _OSI handler.
Obsolete Functions:
    AcpiOsValidateInterface - no longer used.
New Files:
    source/components/utilities/utosi.c

Re-introduced the support to enable multi-byte transfers for Embedded
**Controller (EC) operation regions. A reported problem was found to be a
bug
in the host OS, not in the multi-byte support. Previously, the maximum**

**data
size passed to the EC operation region handler was a single byte. There
are
often EC Fields larger than one byte that need to be transferred, and it
is
useful for the EC driver to lock these as a single transaction. This
change**
*Controller (EC) operation regions. A reported problem was found to be a bug
in the host OS, not in the multi-byte support. Previously, the maximum data
size passed to the EC operation region handler was a single byte. There are
often EC Fields larger than one byte that need to be transferred, and it is
useful for the EC driver to lock these as a single transaction. This change*
enables single transfers larger than 8 bits. This effectively changes the
access to the EC space from ByteAcc to AnyAcc, and will probably require
**changes to the host OS Embedded Controller driver to enable 16/32/64/256-
bit**
changes to the host OS Embedded Controller driver to enable 16/32/64/256-bit
transfers in addition to 8-bit transfers. Alexey Starikovskiy, Lin Ming.

Fixed a problem with the prototype for AcpiOsReadPciConfiguration. The
prototype in acpiosxf.h had the output value pointer as a (void *).
It should be a (UINT64 *). This may affect some host OSL code.

**Fixed a couple problems with the recently modified Linux makefiles for
iASL**
Fixed a couple problems with the recently modified Linux makefiles for iASL
and AcpiExec. These new makefiles place the generated object files in the
**local directory so that there can be no collisions between the files that
are**
local directory so that there can be no collisions between the files that are
shared between them that are compiled with different options.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
**debug version of the code includes the debug output trace mechanism and
has a**
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  88.3K Code, 18.8K Data, 107.1K Total
    Debug Version:     164.0K Code, 51.5K Data, 215.5K Total
  Current Release:
    Non-Debug Version:  89.1K Code, 19.0K Data, 108.1K Total
    Debug Version:     165.1K Code, 51.9K Data, 217.0K Total
```

2) iASL Compiler/Disassembler and Tools:

**iASL/Disassembler: Added a new option (-da, "disassemble all") to load
the**
iASL/Disassembler: Added a new option (-da, "disassemble all") to load the
namespace from and disassemble an entire group of AML files. Useful for
**loading all of the AML tables for a given machine (DSDT, SSDT1...SSDTn)
and**
loading all of the AML tables for a given machine (DSDT, SSDT1...SSDTn) and
disassembling with one simple command. ACPICA BZ 865. Lin Ming.

**iASL: Allow multiple invocations of -e option. This change allows
multiple
uses of -e on the command line: "-e ssdt1.dat -e ssdt2.dat". ACPICA BZ
834.**
*iASL: Allow multiple invocations of -e option. This change allows multiple
uses of -e on the command line: "-e ssdt1.dat -e ssdt2.dat". ACPICA BZ 834.*
Lin Ming.

---

02 July 2010. Summary of changes for version 20100702:

1) ACPI CA Core Subsystem:

Implemented several updates to the recently added GPE reference count support. The model for "wake" GPEs is changing to give the host OS complete control of these GPEs. Eventually, the ACPICA core will not execute any _PRW methods, since the host already must execute them. Also, additional changes were made to help ensure that the reference counts are kept in proper synchronization with reality. Rafael J. Wysocki.

1) Ensure that GPEs are not enabled twice during initialization.
2) Ensure that GPE enable masks stay in sync with the reference count.
3) Do not inadvertently enable GPEs when writing GPE registers.
4) Remove the internal wake reference counter and add new AcpiGpeWakeup interface. This interface will set or clear individual GPEs for wakeup.
5) Remove GpeType argument from AcpiEnable and AcpiDisable. These interfaces are now used for "runtime" GPEs only.

Changed the behavior of the GPE install/remove handler interfaces. The GPE is no longer disabled during this process, as it was found to cause problems on some machines. Rafael J. Wysocki.

Reverted a change introduced in version 20100528 to enable Embedded Controller multi-byte transfers. This change was found to cause problems with Index Fields and possibly Bank Fields. It will be reintroduced when these problems have been resolved.

Fixed a problem with references to Alias objects within Package Objects. A reference to an Alias within the definition of a Package was not always resolved properly. Aliases to objects like Processors, Thermal zones, etc. were resolved to the actual object instead of a reference to the object as it should be. Package objects are only allowed to contain integer, string, buffer, package, and reference objects. Redhat bugzilla 608648.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

  Previous Release:
    Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total

    Debug Version: 164.1K Code, 51.5K Data, 215.6K Total
  Current Release:
    Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total
    Debug Version: 164.0K Code, 51.5K Data, 215.5K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented a new compiler subsystem to allow definition and compilation of the non-AML ACPI tables such as FADT, MADT, SRAT, etc. These are called "ACPI Data Tables", and the new compiler is the "Data Table Compiler". This compiler is intended to simplify the existing error-prone process of creating these tables for the BIOS, as well as allowing the disassembly, modification, recompilation, and override of existing ACPI data tables. See the iASL User Guide for detailed information.

iASL: Implemented a new Template Generator option in support of the new Data Table Compiler. This option will create examples of all known ACPI tables that can be used as the basis for table development. See the iASL documentation and the -T option.

Disassembler and headers: Added support for the WDDT ACPI table (Watchdog Descriptor Table).

Updated the Linux makefiles for iASL and AcpiExec to place the generated object files in the local directory so that there can be no collisions between the shared files between them that are generated with different options.

Added support for Mac OS X in the Unix OSL used for iASL and AcpiExec. Use the #define __APPLE__ to enable this support.

---

28 May 2010. Summary of changes for version 20100528:

Note: The ACPI 4.0a specification was released on April 5, 2010 and is available at www.acpi.info. This is primarily an errata release.

1) ACPI CA Core Subsystem:

Undefined ACPI tables: We are looking for the definitions for the following ACPI tables that have been seen in the field: ATKG, IEIT, GSCI.

Implemented support to enable multi-byte transfers for Embedded Controller (EC) operation regions. Previously, the maximum data size passed to the EC operation region handler was a single byte. There are often EC Fields larger than one byte that need to be transferred, and it is useful for the EC driver to lock these as a single transaction. This change enables single transfers larger than 8 bits. This effectively changes the access to the EC space from ByteAcc to AnyAcc, and will probably require changes to the host OS Embedded

Controller driver to enable 16/32/64/256-bit transfers in addition to 8-bit
Implemented support to enable multi-byte transfers for Embedded Controller (EC) operation regions. Previously, the maximum data size passed to the EC operation region handler was a single byte. There are often EC Fields larger than one byte that need to be transferred, and it is useful for the EC driver to lock these as a single transaction. This change enables single transfers larger than 8 bits. This effectively changes the access to the EC space from ByteAcc to AnyAcc, and will probably require changes to the host OS Embedded Controller driver to enable 16/32/64/256-bit transfers in addition to 8-bit
transfers. Alexey Starikovskiy, Lin Ming

Implemented a performance enhancement for namespace search and access. This change enhances the performance of namespace searches and walks by adding a backpointer to the parent in each namespace node. On large namespaces, this change can improve overall ACPI performance by up to 9X. Adding a pointer to each namespace node increases the overall size of the internal namespace by
Implemented a performance enhancement for namespace search and access. This change enhances the performance of namespace searches and walks by adding a backpointer to the parent in each namespace node. On large namespaces, this change can improve overall ACPI performance by up to 9X. Adding a pointer to each namespace node increases the overall size of the internal namespace by
about 5%, since each namespace entry usually consists of both a namespace node and an ACPI operand object. However, this is the first growth of the namespace in ten years. ACPICA bugzilla 817. Alexey Starikovskiy.

Implemented a performance optimization that reduces the number of namespace walks. On control method exit, only walk the namespace if the method is known to have created namespace objects outside of its local scope. Previously, the entire namespace was traversed on each control method exit. This change can improve overall ACPI performance by up to 3X. Alexey Starikovskiy, Bob Moore.
Implemented a performance optimization that reduces the number of namespace walks. On control method exit, only walk the namespace if the method is known to have created namespace objects outside of its local scope. Previously, the entire namespace was traversed on each control method exit. This change can improve overall ACPI performance by up to 3X. Alexey Starikovskiy, Bob Moore.

Added support to truncate I/O addresses to 16 bits for Windows compatibility.
Added support to truncate I/O addresses to 16 bits for Windows compatibility.
Some ASL code has been seen in the field that inadvertently has bits set **above bit 15. This feature is optional and is enabled if the BIOS requests**
above bit 15. This feature is optional and is enabled if the BIOS requests
any Windows OSI strings. It can also be enabled by the host OS. Matthew Garrett, Bob Moore.

Added support to limit the maximum time for the ASL Sleep() operator. To prevent accidental deep sleeps, limit the maximum time that Sleep() will actually sleep. Configurable, the default maximum is two seconds. ACPICA bugzilla 854.

Added run-time validation support for the _WDG and_WED Microsoft predefined methods. These objects are defined by "Windows Instrumentation", and are not

Added run-time validation support for the _WDG and_WED Microsoft predefined methods. These objects are defined by "Windows Instrumentation", and are not
part of the ACPI spec. ACPICA BZ 860.

Expanded all statistic counters used during namespace and device **initialization from 16 to 32 bits in order to support very large namespaces.**
initialization from 16 to 32 bits in order to support very large namespaces.

Replaced all instances of %d in printf format specifiers with %u since nearly
Replaced all instances of %d in printf format specifiers with %u since nearly
all integers in ACPICA are unsigned.

Fixed the exception namestring for AE_WAKE_ONLY_GPE. Was incorrectly returned
Fixed the exception namestring for AE_WAKE_ONLY_GPE. Was incorrectly returned
as AE_NO_HANDLER.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  88.4K Code, 18.8K Data, 107.2K Total
    Debug Version:     164.2K Code, 51.5K Data, 215.7K Total
  Current Release:
    Non-Debug Version:  88.3K Code, 18.8K Data, 107.1K Total
    Debug Version:     164.1K Code, 51.5K Data, 215.6K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Added compiler support for the _WDG and_WED Microsoft predefined **methods. These objects are defined by "Windows Instrumentation", and are not**
methods. These objects are defined by "Windows Instrumentation", and are not
part of the ACPI spec. ACPICA BZ 860.

AcpiExec: added option to disable the memory tracking mechanism. The -dt option will disable the tracking mechanism, which improves performance considerably.

AcpiExec: Restructured the command line options into -d (disable) and -e (enable) options.

---
28 April 2010. Summary of changes for version 20100428:

1) ACPI CA Core Subsystem:

Implemented GPE support for dynamically loaded ACPI tables. For all GPEs, **including FADT-based and GPE Block Devices, execute any _PRW methods in the**
including FADT-based and GPE Block Devices, execute any _PRW methods in the
new table, and process any _Lxx/_Exx GPE methods in the new table. Any runtime GPE that is referenced by an _Lxx/_Exx method in the new table is immediately enabled. Handles the FADT-defined GPEs as well as GPE Block Devices. Provides compatibility with other ACPI implementations. Two new **files added, evgpeinit.c and evgpeutil.c. ACPICA BZ 833. Lin Ming, Bob Moore.**
files added, evgpeinit.c and evgpeutil.c. ACPICA BZ 833. Lin Ming, Bob Moore.

Fixed a regression introduced in version 20100331 within the table

manager
where initial table loading could fail. This was introduced in the fix
for
AcpiReallocateRootTable. Also, renamed some of fields in the table
manager
Fixed a regression introduced in version 20100331 within the table manager
where initial table loading could fail. This was introduced in the fix for
AcpiReallocateRootTable. Also, renamed some of fields in the table manager
data structures to clarify their meaning and use.

Fixed a possible allocation overrun during internal object copy in
AcpiUtCopySimpleObject. The original code did not correctly handle the
case
where the object to be copied was a namespace node. Lin Ming. ACPICA BZ
847.
AcpiUtCopySimpleObject. The original code did not correctly handle the case
where the object to be copied was a namespace node. Lin Ming. ACPICA BZ 847.

Updated the allocation dump routine, AcpiUtDumpAllocation and fixed a
possible access beyond end-of-allocation. Also, now fully validate
descriptor
possible access beyond end-of-allocation. Also, now fully validate descriptor
(size and type) before output. Lin Ming, Bob Moore. ACPICA BZ 847

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  87.9K Code, 18.6K Data, 106.5K Total
    Debug Version:     163.5K Code, 51.3K Data, 214.8K Total
  Current Release:
    Non-Debug Version:  88.4K Code, 18.8K Data, 107.2K Total
    Debug Version:     164.2K Code, 51.5K Data, 215.7K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented Min/Max/Len/Gran validation for address resource
descriptors. This change implements validation for the address fields
that
descriptors. This change implements validation for the address fields that
are common to all address-type resource descriptors. These checks are
implemented: Checks for valid Min/Max, length within the Min/Max window,
valid granularity, Min/Max a multiple of granularity, and _MIF/_MAF as
per
table 6-40 in the ACPI 4.0a specification. Also split the large
aslrestype1.c
valid granularity, Min/Max a multiple of granularity, and _MIF/_MAF as per
table 6-40 in the ACPI 4.0a specification. Also split the large aslrestype1.c
and aslrestype2.c files into five new files. ACPICA BZ 840.

iASL: Added support for the _Wxx predefined names. This support was
missing
iASL: Added support for the _Wxx predefined names. This support was missing
and these names were not recognized by the compiler as valid predefined
names. ACPICA BZ 851.

iASL: Added an error for all predefined names that are defined to return
no
value and thus must be implemented as Control Methods. These include all
of
iASL: Added an error for all predefined names that are defined to return no
value and thus must be implemented as Control Methods. These include all of

the _Lxx, _Exx, _Wxx, and _Qxx names, as well as some other miscellaneous
names such as _DIS, _INI, _IRC, _OFF, _ON, and _PSx. ACPICA BZ 850, 856.

iASL: Implemented the -ts option to emit hex AML data in ASL format, as
an
ASL Buffer. Allows ACPI tables to be easily included within ASL files, to
be
dynamically loaded via the Load() operator. Also cleaned up output for
the
-
iASL: Implemented the -ts option to emit hex AML data in ASL format, as an
ASL Buffer. Allows ACPI tables to be easily included within ASL files, to be
dynamically loaded via the Load() operator. Also cleaned up output for the -
ta and -tc options. ACPICA BZ 853.

Tests: Added a new file with examples of extended iASL error checking.
Demonstrates the advanced error checking ability of the iASL compiler.
Available at tests/misc/badcode.asl.

---
31 March 2010. Summary of changes for version 20100331:

1) ACPI CA Core Subsystem:

Completed a major update for the GPE support in order to improve support
for
shared GPEs and to simplify both host OS and ACPICA code. Added a
reference
count mechanism to support shared GPEs that require multiple device
drivers.
Completed a major update for the GPE support in order to improve support for
shared GPEs and to simplify both host OS and ACPICA code. Added a reference
count mechanism to support shared GPEs that require multiple device drivers.
Several external interfaces have changed. One external interface has been
removed. One new external interface was added. Most of the GPE external
interfaces now use the GPE spinlock instead of the events mutex (and the
Flags parameter for many GPE interfaces has been removed.) See the
updated
ACPICA Programmer Reference for details. Matthew Garrett, Bob Moore,
Rafael
Flags parameter for many GPE interfaces has been removed.) See the updated
ACPICA Programmer Reference for details. Matthew Garrett, Bob Moore, Rafael
Wysocki. ACPICA BZ 831.

```
Changed:
    AcpiEnableGpe, AcpiDisableGpe, AcpiClearGpe, AcpiGetGpeStatus
Removed:
    AcpiSetGpeType
New:
    AcpiSetGpe
```

Implemented write support for DataTable operation regions. These regions
are
defined via the DataTableRegion() operator. Previously, only read support
was
implemented. The ACPI specification allows DataTableRegions to be
read/write,
Implemented write support for DataTable operation regions. These regions are
defined via the DataTableRegion() operator. Previously, only read support was
implemented. The ACPI specification allows DataTableRegions to be read/write,
however.

Implemented a new subsystem option to force a copy of the DSDT to local
memory. Optionally copy the entire DSDT to local memory (instead of
simply
mapping it.) There are some (albeit very rare) BIOSs that corrupt or

replace
the original DSDT, creating the need for this option. Default is FALSE,
do
memory. Optionally copy the entire DSDT to local memory (instead of simply
mapping it.) There are some (albeit very rare) BIOSs that corrupt or replace
the original DSDT, creating the need for this option. Default is FALSE, do
not copy the DSDT.

Implemented detection of a corrupted or replaced DSDT. This change adds
support to detect a DSDT that has been corrupted and/or replaced from
outside
the OS (by firmware). This is typically catastrophic for the system, but
has
support to detect a DSDT that has been corrupted and/or replaced from outside
the OS (by firmware). This is typically catastrophic for the system, but has
been seen on some machines. Once this problem has been detected, the DSDT
copy option can be enabled via system configuration. Lin Ming, Bob Moore.

Fixed two problems with AcpiReallocateRootTable during the root table
copy.
Fixed two problems with AcpiReallocateRootTable during the root table copy.
When copying the root table to the new allocation, the length used was
incorrect. The new size was used instead of the current table size,
meaning
too much data was copied. Also, the count of available slots for ACPI
tables
incorrect. The new size was used instead of the current table size, meaning
too much data was copied. Also, the count of available slots for ACPI tables
was not set correctly. Alexey Starikovskiy, Bob Moore.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  87.5K Code, 18.4K Data, 105.9K Total
    Debug Version:     163.4K Code, 51.1K Data, 214.5K Total
  Current Release:
    Non-Debug Version:  87.9K Code, 18.6K Data, 106.5K Total
    Debug Version:     163.5K Code, 51.3K Data, 214.8K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implement limited typechecking for values returned from predefined
control methods. The type of any returned static (unnamed) object is now
validated. For example, Return(1). ACPICA BZ 786.

iASL: Fixed a predefined name object verification regression. Fixes a
problem
iASL: Fixed a predefined name object verification regression. Fixes a problem
introduced in version 20100304. An error is incorrectly generated if a
predefined name is declared as a static named object with a value defined
using the keywords "Zero", "One", or "Ones". Lin Ming.

iASL: Added Windows 7 support for the -g option (get local ACPI tables)
by
iASL: Added Windows 7 support for the -g option (get local ACPI tables) by
reducing the requested registry access rights. ACPICA BZ 842.

Disassembler: fixed a possible fault when generating External()
statements.
Introduced in commit ae7d6fd: Properly handle externals with parent-
prefix

Disassembler: fixed a possible fault when generating External() statements.
Introduced in commit ae7d6fd: Properly handle externals with parent-prefix
(carat). Fixes a string length allocation calculation. Lin Ming.

---
04 March 2010. Summary of changes for version 20100304:

1) ACPI CA Core Subsystem:

Fixed a possible problem with the AML Mutex handling function
AcpiExReleaseMutex where the function could fault under the very rare
condition when the interpreter has blocked, the interpreter lock is
released,
condition when the interpreter has blocked, the interpreter lock is released,
the interpreter is then reentered via the same thread, and attempts to
acquire an AML mutex that was previously acquired. FreeBSD report 140979.
Lin
acquire an AML mutex that was previously acquired. FreeBSD report 140979. Lin
Ming.

Implemented additional configuration support for the AML "Debug Object".
Output from the debug object can now be enabled via a global variable,
AcpiGbl_EnableAmlDebugObject. This will assist with remote machine
debugging.
This debug output is now available in the release version of ACPICA
instead
of just the debug version. Also, the entire debug output module can now
be
AcpiGbl_EnableAmlDebugObject. This will assist with remote machine debugging.
This debug output is now available in the release version of ACPICA instead
of just the debug version. Also, the entire debug output module can now be
configured out of the ACPICA build if desired. One new file added,
executer/exdebug.c. Lin Ming, Bob Moore.

Added header support for the ACPI MCHI table (Management Controller Host
Interface Table). This table was added in ACPI 4.0, but the defining
document
Interface Table). This table was added in ACPI 4.0, but the defining document
has only recently become available.

Standardized output of integer values for ACPICA warnings/errors. Always
use
0x prefix for hex output, always use %u for unsigned integer decimal
output.
Affects ACPI_INFO, ACPI_ERROR, ACPI_EXCEPTION, and ACPI_WARNING (about
400
Standardized output of integer values for ACPICA warnings/errors. Always use
0x prefix for hex output, always use %u for unsigned integer decimal output.
Affects ACPI_INFO, ACPI_ERROR, ACPI_EXCEPTION, and ACPI_WARNING (about 400
invocations.) These invocations were converted from the original
ACPI_DEBUG_PRINT invocations and were not consistent. ACPICA BZ 835.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  87.1K Code, 18.0K Data, 105.1K Total
    Debug Version:     163.5K Code, 50.9K Data, 214.4K Total
  Current Release:
    Non-Debug Version:  87.5K Code, 18.4K Data, 105.9K Total
    Debug Version:     163.4K Code, 51.1K Data, 214.5K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented typechecking support for static (non-control method)
predefined named objects that are declared with the Name() operator. For
example, the type of this object is now validated to be of type Integer:
Name(_BBN, 1). This change migrates the compiler to using the core
predefined
Name(_BBN, 1). This change migrates the compiler to using the core predefined
name table instead of maintaining a local version. Added a new file,
aslpredef.c. ACPICA BZ 832.

Disassembler: Added support for the ACPI 4.0 MCHI table.

---
21 January 2010. Summary of changes for version 20100121:

1) ACPI CA Core Subsystem:

Added the 2010 copyright to all module headers and signons. This affects
virtually every file in the ACPICA core subsystem, the iASL compiler, the
tools/utilities, and the test suites.

Implemented a change to the AcpiGetDevices interface to eliminate
unnecessary
Implemented a change to the AcpiGetDevices interface to eliminate unnecessary
invocations of the _STA method. In the case where a specific _HID is
requested, do not run _STA until a _HID match is found. This eliminates
potentially dozens of _STA calls during a search for a particular
device/HID,
potentially dozens of _STA calls during a search for a particular device/HID,
which in turn can improve boot times. ACPICA BZ 828. Lin Ming.

Implemented an additional repair for predefined method return values.
Attempt
to repair unexpected NULL elements within returned Package objects.
Create
an
Integer of value zero, a NULL String, or a zero-length Buffer as
appropriate.
Implemented an additional repair for predefined method return values. Attempt
to repair unexpected NULL elements within returned Package objects. Create an
Integer of value zero, a NULL String, or a zero-length Buffer as appropriate.
ACPICA BZ 818. Lin Ming, Bob Moore.

Removed the obsolete ACPI_INTEGER data type. This type was introduced as
the
code was migrated from ACPI 1.0 (with 32-bit AML integers) to ACPI 2.0
(with
64-bit AML integers). It is now obsolete and this change removes it from
the
ACPICA code base, replaced by UINT64. The original typedef has been
retained
for now for compatibility with existing device driver code. ACPICA BZ
824.
Removed the obsolete ACPI_INTEGER data type. This type was introduced as the
code was migrated from ACPI 1.0 (with 32-bit AML integers) to ACPI 2.0 (with
64-bit AML integers). It is now obsolete and this change removes it from the
ACPICA code base, replaced by UINT64. The original typedef has been retained
for now for compatibility with existing device driver code. ACPICA BZ 824.

Removed the unused UINT32_STRUCT type, and the obsolete Integer64 field
in
Removed the unused UINT32_STRUCT type, and the obsolete Integer64 field in
the parse tree object.

Added additional warning options for the gcc-4 generation. Updated the

source
accordingly. This includes some code restructuring to eliminate
unreachable
code, elimination of some gotos, elimination of unused return values,
some
Added additional warning options for the gcc-4 generation. Updated the source
accordingly. This includes some code restructuring to eliminate unreachable
code, elimination of some gotos, elimination of unused return values, some
additional casting, and removal of redundant declarations.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  87.0K Code, 18.0K Data, 105.0K Total
    Debug Version:     163.4K Code, 50.8K Data, 214.2K Total
  Current Release:
    Non-Debug Version:  87.1K Code, 18.0K Data, 105.1K Total
    Debug Version:     163.5K Code, 50.9K Data, 214.4K Total
```

2) iASL Compiler/Disassembler and Tools:

No functional changes for this release.

---
14 December 2009. Summary of changes for version 20091214:

1) ACPI CA Core Subsystem:

Enhanced automatic data type conversions for predefined name repairs.
This
change expands the automatic repairs/conversions for predefined name
return
values to make Integers, Strings, and Buffers fully interchangeable.
Also,
a
Buffer can be converted to a Package of Integers if necessary. The
nsrepair.c
Enhanced automatic data type conversions for predefined name repairs. This
change expands the automatic repairs/conversions for predefined name return
values to make Integers, Strings, and Buffers fully interchangeable. Also, a
Buffer can be converted to a Package of Integers if necessary. The nsrepair.c
module was completely restructured. Lin Ming, Bob Moore.

Implemented automatic removal of null package elements during predefined
name
Implemented automatic removal of null package elements during predefined name
repairs. This change will automatically remove embedded and trailing NULL
package elements from returned package objects that are defined to
contain
a
variable number of sub-packages. The driver is then presented with a
package
package elements from returned package objects that are defined to contain a
variable number of sub-packages. The driver is then presented with a package
with no null elements to deal with. ACPICA BZ 819.

Implemented a repair for the predefined _FDE and _GTM names. The expected
return value for both names is a Buffer of 5 DWORDs. This repair fixes
two
possible problems (both seen in the field), where a package of integers
is

returned, or a buffer of BYTEs is returned. With assistance from Jung-uk Kim.
return value for both names is a Buffer of 5 DWORDs. This repair fixes two possible problems (both seen in the field), where a package of integers is returned, or a buffer of BYTEs is returned. With assistance from Jung-uk Kim.

Implemented additional module-level code support. This change will properly execute module-level code that is not at the root of the namespace (under a Device object, etc.). Now executes the code within the current scope instead
Implemented additional module-level code support. This change will properly execute module-level code that is not at the root of the namespace (under a Device object, etc.). Now executes the code within the current scope instead of the root. ACPICA BZ 762. Lin Ming.

Fixed possible mutex acquisition errors when running _REG methods. Fixes a problem where mutex errors can occur when running a _REG method that is in the same scope as a method-defined operation region or an operation region under a module-level IF block. This type of code is rare, so the problem has
Fixed possible mutex acquisition errors when running _REG methods. Fixes a problem where mutex errors can occur when running a _REG method that is in the same scope as a method-defined operation region or an operation region under a module-level IF block. This type of code is rare, so the problem has not been seen before. ACPICA BZ 826. Lin Ming, Bob Moore.

Fixed a possible memory leak during module-level code execution. An object
Fixed a possible memory leak during module-level code execution. An object could be leaked for each block of executed module-level code if the **interpreter slack mode is enabled This change deletes any implicitly returned**
interpreter slack mode is enabled This change deletes any implicitly returned object from the module-level code block. Lin Ming.

Removed messages for successful predefined repair(s). The repair mechanism was considered too wordy. Now, messages are only unconditionally emitted if
Removed messages for successful predefined repair(s). The repair mechanism was considered too wordy. Now, messages are only unconditionally emitted if the return object cannot be repaired. Existing messages for successful **repairs were converted to ACPI_DEBUG_PRINT messages for now. ACPICA BZ 827.**
repairs were converted to ACPI_DEBUG_PRINT messages for now. ACPICA BZ 827.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  86.6K Code, 18.2K Data, 104.8K Total
    Debug Version:     162.7K Code, 50.8K Data, 213.5K Total
  Current Release:
    Non-Debug Version:  87.0K Code, 18.0K Data, 105.0K Total
    Debug Version:     163.4K Code, 50.8K Data, 214.2K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a regression introduced in 20091112 where intermediate .SRC files
iASL: Fixed a regression introduced in 20091112 where intermediate .SRC files were no longer automatically removed at the termination of the compile.

acpiexec: Implemented the -f option to specify default region fill value. This option specifies the value used to initialize buffers that simulate **operation regions. Default value is zero. Useful for debugging problems that**
operation regions. Default value is zero. Useful for debugging problems that depend on a specific initial value for a region or field.

---
12 November 2009. Summary of changes for version 20091112:

1) ACPI CA Core Subsystem:

Implemented a post-order callback to AcpiWalkNamespace. The existing interface only has a pre-order callback. This change adds an additional **parameter for a post-order callback which will be more useful for bus scans.**
parameter for a post-order callback which will be more useful for bus scans. ACPICA BZ 779. Lin Ming. Updated the ACPICA Programmer Reference.

Modified the behavior of the operation region memory mapping cache for **SystemMemory. Ensure that the memory mappings created for operation regions**
SystemMemory. Ensure that the memory mappings created for operation regions do not cross 4K page boundaries. Crossing a page boundary while mapping **regions can cause kernel warnings on some hosts if the pages have different attributes. Such regions are probably BIOS bugs, and this is the workaround.**
regions can cause kernel warnings on some hosts if the pages have different attributes. Such regions are probably BIOS bugs, and this is the workaround. Linux BZ 14445. Lin Ming.

Implemented an automatic repair for predefined methods that must return **sorted lists. This change will repair (by sorting) packages returned by _ALR, _PSS, and _TSS. Drivers can now assume that the packages are correctly sorted**
sorted lists. This change will repair (by sorting) packages returned by _ALR, _PSS, and _TSS. Drivers can now assume that the packages are correctly sorted and do not contain NULL package elements. Adds one new file, namespace/nsrepair2.c. ACPICA BZ 784. Lin Ming, Bob Moore.

Fixed a possible fault during predefined name validation if a return Package
Fixed a possible fault during predefined name validation if a return Package object contains NULL elements. Also adds a warning if a NULL element is **followed by any non-null elements. ACPICA BZ 813, 814. Future enhancement may**
followed by any non-null elements. ACPICA BZ 813, 814. Future enhancement may include repair or removal of all such NULL elements where possible.

Implemented additional module-level executable AML code support. This change
Implemented additional module-level executable AML code support. This change will execute module-level code that is not at the root of the namespace **(under a Device object, etc.) at table load time. Module-level executable AML**
(under a Device object, etc.) at table load time. Module-level executable AML code has been illegal since ACPI 2.0. ACPICA BZ 762. Lin Ming.

Implemented a new internal function to create Integer objects. This function
Implemented a new internal function to create Integer objects. This function simplifies miscellaneous object creation code. ACPICA BZ 823.

Reduced the severity of predefined repair messages, Warning to Info. Since the object was successfully repaired, a warning is too severe. Reduced to an info message for now. These messages may eventually be changed to debug-only.
Reduced the severity of predefined repair messages, Warning to Info. Since the object was successfully repaired, a warning is too severe. Reduced to an info message for now. These messages may eventually be changed to debug-only. ACPICA BZ 812.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  85.8K Code, 18.0K Data, 103.8K Total
    Debug Version:     161.8K Code, 50.6K Data, 212.4K Total
  Current Release:
    Non-Debug Version:  86.6K Code, 18.2K Data, 104.8K Total
    Debug Version:     162.7K Code, 50.8K Data, 213.5K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Implemented Switch() with While(1) so that Break works correctly. This change correctly implements the Switch operator with a surrounding While(1)
iASL: Implemented Switch() with While(1) so that Break works correctly. This change correctly implements the Switch operator with a surrounding While(1) so that the Break operator works as expected. ACPICA BZ 461. Lin Ming.

iASL: Added a message if a package initializer list is shorter than package length. Adds a new remark for a Package() declaration if an initializer list
iASL: Added a message if a package initializer list is shorter than package length. Adds a new remark for a Package() declaration if an initializer list exists, but is shorter than the declared length of the package. Although technically legal, this is probably a coding error and it is seen in the field. ACPICA BZ 815. Lin Ming, Bob Moore.

iASL: Fixed a problem where the compiler could fault after the maximum number
iASL: Fixed a problem where the compiler could fault after the maximum number of errors was reached (200).

acpixtract: Fixed a possible warning for pointer cast if the compiler warning
acpixtract: Fixed a possible warning for pointer cast if the compiler warning level set very high.

---

13 October 2009. Summary of changes for version 20091013:

1) ACPI CA Core Subsystem:

Fixed a problem where an Operation Region _REG method could be executed more than once. If a custom address space handler is installed by the host before the "initialize operation regions" phase of the ACPICA initialization, any
Fixed a problem where an Operation Region _REG method could be executed more than once. If a custom address space handler is installed by the host before the "initialize operation regions" phase of the ACPICA initialization, any _REG methods for that address space could be executed twice. This change fixes the problem. ACPICA BZ 427. Lin Ming.

Fixed a possible memory leak for the Scope() ASL operator. When the exact invocation of "Scope(\)" is executed (change scope to root), one internal operand object was leaked. Lin Ming.

Implemented a run-time repair for the _MAT predefined method. If the _MAT return value is defined as a Field object in the AML, and the field **size is less than or equal to the default width of an integer (32 or 64),_MAT**
size is less than or equal to the default width of an integer (32 or 64),_MAT can incorrectly return an Integer instead of a Buffer. ACPICA now automatically repairs this problem. ACPICA BZ 810.

Implemented a run-time repair for the _BIF and _BIX predefined methods. The
Implemented a run-time repair for the _BIF and _BIX predefined methods. The "OEM Information" field is often incorrectly returned as an Integer with **value zero if the field is not supported by the platform. This is due to an**
value zero if the field is not supported by the platform. This is due to an ambiguity in the ACPI specification. The field should always be a string. ACPICA now automatically repairs this problem by returning a NULL string within the returned Package. ACPICA BZ 807.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  85.6K Code, 18.0K Data, 103.6K Total
    Debug Version:     161.7K Code, 50.9K Data, 212.6K Total
  Current Release:
    Non-Debug Version:  85.8K Code, 18.0K Data, 103.8K Total
    Debug Version:     161.8K Code, 50.6K Data, 212.4K Total
```

2) iASL Compiler/Disassembler and Tools:

Disassembler: Fixed a problem where references to external symbols that **contained one or more parent-prefixes (carats) were not handled correctly,**
contained one or more parent-prefixes (carats) were not handled correctly, possibly causing a fault. ACPICA BZ 806. Lin Ming.

Disassembler: Restructured the code so that all functions that handle external symbols are in a single module. One new file is added, common/dmextern.c.

AML Debugger: Added a max count argument for the Batch command (which executes multiple predefined methods within the namespace.)

iASL: Updated the compiler documentation (User Reference.) Available at http://www.acpica.org/documentation/. ACPICA BZ 750.

AcpiXtract: Updated for Lint and other formatting changes. Close all open files.

---

03 September 2009. Summary of changes for version 20090903:

1) ACPI CA Core Subsystem:

For Windows Vista compatibility, added the automatic execution of an _INI method located at the namespace root (_INI). This method is executed at table load time. This support is in addition to the automatic execution of *table load time. This support is in addition to the automatic execution of* _SB._INI. Lin Ming.

Fixed a possible memory leak in the interpreter for AML package objects if the package initializer list is longer than the defined size of the package. This apparently can only happen if the BIOS changes the package size on the *Fixed a possible memory leak in the interpreter for AML package objects if the package initializer list is longer than the defined size of the package. This apparently can only happen if the BIOS changes the package size on the* fly (seen in a _PSS object), as ASL compilers do not allow this. The interpreter will truncate the package to the defined size (and issue an error message), but previously could leave the extra objects undeleted if they were pre-created during the argument processing (such is the case if the package *interpreter will truncate the package to the defined size (and issue an error message), but previously could leave the extra objects undeleted if they were pre-created during the argument processing (such is the case if the package* consists of a number of sub-packages as in the _PSS.) ACPICA BZ 805.

Fixed a problem seen when a Buffer or String is stored to itself via ASL. This has been reported in the field. Previously, ACPICA would zero out the *This has been reported in the field. Previously, ACPICA would zero out the* buffer/string. Now, the operation is treated as a noop. Provides Windows compatibility. ACPICA BZ 803. Lin Ming.

Removed an extraneous error message for ASL constructs of the form Store(LocalX,LocalX) when LocalX is uninitialized. These curious statements are seen in many BIOSs and are once again treated as NOOPs and no error is *Store(LocalX,LocalX) when LocalX is uninitialized. These curious statements are seen in many BIOSs and are once again treated as NOOPs and no error is* emitted when they are encountered. ACPICA BZ 785.

Fixed an extraneous warning message if a _DSM reserved method returns a Package object. _DSM can return any type of object, so validation on the return type cannot be performed. ACPICA BZ 802.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a *debug version of the code includes the debug output trace mechanism and has a* much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  85.5K Code, 18.0K Data, 103.5K Total
    Debug Version:     161.6K Code, 50.9K Data, 212.5K Total
```



```
  Current Release:
    Non-Debug Version:  85.6K Code, 18.0K Data, 103.6K Total
    Debug Version:     161.7K Code, 50.9K Data, 212.6K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a problem with the use of the Alias operator and Resource Templates. The correct alias is now constructed and no error is emitted. ACPICA BZ 738.

iASL: Implemented the -I option to specify additional search directories for *iASL: Implemented the -I option to specify additional search directories for* include files. Allows multiple additional search paths for include files. Directories are searched in the order specified on the command line (after *Directories are searched in the order specified on the command line (after* the local directory is searched.) ACPICA BZ 800.

iASL: Fixed a problem where the full pathname for include files was not emitted for warnings/errors. This caused the IDE support to not work properly. ACPICA BZ 765.

iASL: Implemented the -@ option to specify a Windows-style response file containing additional command line options. ACPICA BZ 801.

AcpiExec: Added support to load multiple AML files simultaneously (such as a *AcpiExec: Added support to load multiple AML files simultaneously (such as a* DSDT and multiple SSDTs). Also added support for wildcards within the AML pathname. These features allow all machine tables to be easily loaded and debugged together. ACPICA BZ 804.

Disassembler: Added missing support for disassembly of HEST table Error Bank *Disassembler: Added missing support for disassembly of HEST table Error Bank* subtables.

---

30 July 2009. Summary of changes for version 20090730:

The ACPI 4.0 implementation for ACPICA is complete with this release.

1) ACPI CA Core Subsystem:

ACPI 4.0: Added header file support for all new and changed ACPI tables. Completely new tables are: IBFT, IVRS, MSCT, and WAET. Tables that are new for ACPI 4.0, but have previously been supported in ACPICA are: CPEP, BERT, EINJ, ERST, and HEST. Other newly supported tables are: UEFI and WDAT. There *Completely new tables are: IBFT, IVRS, MSCT, and WAET. Tables that are new for ACPI 4.0, but have previously been supported in ACPICA are: CPEP, BERT, EINJ, ERST, and HEST. Other newly supported tables are: UEFI and WDAT. There* have been some ACPI 4.0 changes to other existing tables. Split the large actbl1.h header into the existing actbl2.h header. ACPICA BZ 774.

ACPI 4.0: Implemented predefined name validation for all new names. There are 31 new names in ACPI 4.0. The predefined validation module was split into two *ACPI 4.0: Implemented predefined name validation for all new names. There are 31 new names in ACPI 4.0. The predefined validation module was split into two* files. The new file is namespace/nsrepair.c. ACPICA BZ 770.

Implemented support for so-called "module-level executable code". This is executable AML code that exists outside of any control method and is intended to be executed at table load time. Although illegal since ACPI 2.0, this type of code still exists and is apparently still being created. Blocks of this code are now detected and executed as intended. Currently, the code blocks must exist under either an If, Else, or While construct; these are the typical cases seen in the field. ACPICA BZ 762. Lin Ming.

Implemented an automatic dynamic repair for predefined names that return nested Package objects. This applies to predefined names that are defined to return a variable-length Package of sub-packages. If the number of sub-packages is one, BIOS code is occasionally seen that creates a simple single package with no sub-packages. This code attempts to fix the problem by wrapping a new package object around the existing package. These methods can be repaired: _ALR, _CSD, _HPX, _MLS, _PRT, _PSS, _TRT, and _TSS. ACPICA BZ 790.

Fixed a regression introduced in 20090625 for the AcpiGetDevices interface. The _HID/_CID matching was broken and no longer matched IDs correctly. ACPICA BZ 793.

Fixed a problem with AcpiReset where the reset would silently fail if the register was one of the protected I/O ports. AcpiReset now bypasses the port validation mechanism. This may eventually be driven into the AcpiRead/Write interfaces.

Fixed a regression related to the recent update of the AcpiRead/Write interfaces. A sleep/suspend could fail if the optional PM2 Control register does not exist during an attempt to write the Bus Master Arbitration bit. (However, some hosts already delete the code that writes this bit, and the code may in fact be obsolete at this date.) ACPICA BZ 799.

Fixed a problem where AcpiTerminate could fault if inadvertently called twice in succession. ACPICA BZ 795.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  84.7K Code, 17.8K Data, 102.5K Total
    Debug Version:     160.5K Code, 50.6K Data, 211.1K Total
  Current Release:
    Non-Debug Version:  85.5K Code, 18.0K Data, 103.5K Total
    Debug Version:     161.6K Code, 50.9K Data, 212.5K Total
```

2) iASL Compiler/Disassembler and Tools:

ACPI 4.0: Implemented disassembler support for all new ACPI tables and changes to existing tables. ACPICA BZ 775.

---

25 June 2009. Summary of changes for version 20090625:

The ACPI 4.0 Specification was released on June 16 and is available at www.acpi.info. ACPICA implementation of ACPI 4.0 is underway and will continue for the next few releases.

1) ACPI CA Core Subsystem:

ACPI 4.0: Implemented interpreter support for the IPMI operation region address space. Includes support for bi-directional data buffers and an IPMI address space handler (to be installed by an IPMI device driver.) ACPICA BZ 773. Lin Ming.

ACPI 4.0: Added changes for existing ACPI tables - FACS and SRAT. Includes support in both the header files and the disassembler.

Completed a major update for the AcpiGetObjectInfo external interface. Changes include:

- Support for variable, unlimited length HID, UID, and CID strings.
- Support Processor objects the same as Devices (HID,UID,CID,ADR,STA, etc.)
- Call the _SxW power methods on behalf of a device object.
- Determine if a device is a PCI root bridge.
- Change the ACPI_BUFFER parameter to ACPI_DEVICE_INFO.

These changes will require an update to all callers of this interface. See the updated ACPICA Programmer Reference for details. One new source file has been added - utilities/utids.c. ACPICA BZ 368, 780.

Updated the AcpiRead and AcpiWrite external interfaces to support 64-bit transfers. The Value parameter has been extended from 32 bits to 64 bits in order to support new ACPI 4.0 tables. These changes will require an update to

order to support new ACPI 4.0 tables. These changes will require an update to all callers of these interfaces. See the ACPICA Programmer Reference for details. ACPICA BZ 768.

Fixed several problems with AcpiAttachData. The handler was not invoked when the host node was deleted. The data sub-object was not automatically deleted when the host node was deleted. The interface to the handler had an unused parameter, this was removed. ACPICA BZ 778.

Enhanced the function that dumps ACPI table headers. All non-printable characters in the string fields are now replaced with '?' (Signature, OemId, OemTableId, and CompilerId.) ACPI tables with non-printable characters in these fields are occasionally seen in the field. ACPICA BZ 788.

Fixed a problem with predefined method repair code where the code that attempts to repair/convert an object of incorrect type is only executed on the first time the predefined method is called. The mechanism that disables warnings on subsequent calls was interfering with the repair mechanism. ACPICA BZ 781.

Fixed a possible memory leak in the predefined validation/repair code when a buffer is automatically converted to an expected string object.

Removed obsolete 16-bit files from the distribution and from the current git tree head. ACPICA BZ 776.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  83.4K Code, 17.5K Data, 100.9K Total
    Debug Version:     158.9K Code, 50.0K Data, 208.9K Total
  Current Release:
    Non-Debug Version:  84.7K Code, 17.8K Data, 102.5K Total
    Debug Version:     160.5K Code, 50.6K Data, 211.1K Total
```

2) iASL Compiler/Disassembler and Tools:

ACPI 4.0: iASL and Disassembler - implemented support for the new IPMI operation region keyword. ACPICA BZ 771, 772. Lin Ming.

ACPI 4.0: iASL - implemented compile-time validation support for all new predefined names and control methods (31 total). ACPICA BZ 769.

---

21 May 2009. Summary of changes for version 20090521:

1) ACPI CA Core Subsystem:

Disabled the preservation of the SCI enable bit in the PM1 control register. The SCI enable bit (bit 0, SCI_EN) is defined by the ACPI specification to be a "preserved" bit - "OSPM always preserves this bit position", section 4.7.3.2.1. However, some machines fail if this bit is in fact preserved because the bit needs to be explicitly set by the OS as a workaround. No machines fail if the bit is not preserved. Therefore, ACPICA no longer attempts to preserve this bit.

Fixed a problem in AcpiRsGetPciRoutingTableLength where an invalid or incorrectly formed _PRT package could cause a fault. Added validation to ensure that each package element is actually a sub-package.

Implemented a new interface to install or override a single control method, AcpiInstallMethod. This interface is useful when debugging in order to repair an existing method or to install a missing method without having to override the entire ACPI table. See the ACPICA Programmer Reference for use and examples. Lin Ming, Bob Moore.

Fixed several reference count issues with the DdbHandle object that is created from a Load or LoadTable operator. Prevent premature deletion of the object. Also, mark the object as invalid once the table has been unloaded. This is needed because the handle itself may not be deleted after the table unload, depending on whether it has been stored in a named object by the caller. Lin Ming.

Fixed a problem with Mutex Sync Levels. Fixed a problem where if multiple mutexes of the same sync level are acquired but then not released in strict opposite order, the internally maintained Current Sync Level becomes confused and can cause subsequent execution errors. ACPICA BZ 471.

Changed the allowable release order for ASL mutex objects. The ACPI 4.0 specification has been changed to make the SyncLevel for mutex objects more useful. When releasing a mutex, the SyncLevel of the mutex must now be the same as the current sync level. This makes more sense than the previous rule (SyncLevel less than or equal). This change updates the code to match the

specification.

Fixed a problem with the local version of the AcpiOsPurgeCache function. The
Fixed a problem with the local version of the AcpiOsPurgeCache function. The
(local) cache must be locked during all cache object deletions. Andrew
Baumann.

Updated the Load operator to use operation region interfaces. This replaces
direct memory mapping with region access calls. Now, all region accesses go
Updated the Load operator to use operation region interfaces. This replaces
direct memory mapping with region access calls. Now, all region accesses go
through the installed region handler as they should.

Simplified and optimized the NsGetNextNode function. Reduced parameter count
Simplified and optimized the NsGetNextNode function. Reduced parameter count
and reduced code for this frequently used function.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  82.8K Code, 17.5K Data, 100.3K Total
    Debug Version:     158.0K Code, 49.9K Data, 207.9K Total
  Current Release:
    Non-Debug Version:  83.4K Code, 17.5K Data, 100.9K Total
    Debug Version:     158.9K Code, 50.0K Data, 208.9K Total
```

2) iASL Compiler/Disassembler and Tools:

Disassembler: Fixed some issues with DMAR, HEST, MADT tables. Some problems
with sub-table disassembly and handling invalid sub-tables. Attempt recovery
Disassembler: Fixed some issues with DMAR, HEST, MADT tables. Some problems
with sub-table disassembly and handling invalid sub-tables. Attempt recovery
after an invalid sub-table ID.

---
22 April 2009. Summary of changes for version 20090422:

1) ACPI CA Core Subsystem:

Fixed a compatibility issue with the recently released I/O port protection
Fixed a compatibility issue with the recently released I/O port protection
mechanism. For windows compatibility, 1) On a port protection violation,
simply ignore the request and do not return an exception (allow the control
simply ignore the request and do not return an exception (allow the control
method to continue execution.) 2) If only part of the request overlaps a
protected port, read/write the individual ports that are not protected. Linux
protected port, read/write the individual ports that are not protected. Linux
BZ 13036. Lin Ming

Enhanced the execution of the ASL/AML BreakPoint operator so that it actually
Enhanced the execution of the ASL/AML BreakPoint operator so that it actually

breaks into the AML debugger if the debugger is present. This matches the
ACPI-defined behavior.

Fixed several possible warnings related to the use of the configurable
ACPI_THREAD_ID. This type can now be configured as either an integer or a
pointer with no warnings. Also fixes several warnings in printf-like
statements for the 64-bit build when the type is configured as a pointer.
ACPICA BZ 766, 767.

Fixed a number of possible warnings when compiling with gcc 4+ (depending on
warning options.) Examples include printf formats, aliasing, unused globals,
Fixed a number of possible warnings when compiling with gcc 4+ (depending on
warning options.) Examples include printf formats, aliasing, unused globals,
missing prototypes, missing switch default statements, use of non-ANSI
library functions, use of non-ANSI constructs. See generate/unix/Makefile for
library functions, use of non-ANSI constructs. See generate/unix/Makefile for
a list of warning options used with gcc 3 and 4. ACPICA BZ 735.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  82.6K Code, 17.6K Data, 100.2K Total
    Debug Version:     157.7K Code, 49.9K Data, 207.6K Total
  Current Release:
    Non-Debug Version:  82.8K Code, 17.5K Data, 100.3K Total
    Debug Version:     158.0K Code, 49.9K Data, 207.9K Total
```

2) iASL Compiler/Disassembler and Tools:

iASL: Fixed a generation warning from Bison 2.3 and fixed several warnings on
iASL: Fixed a generation warning from Bison 2.3 and fixed several warnings on
the 64-bit build.

iASL: Fixed a problem where the Unix/Linux versions of the compiler could not
iASL: Fixed a problem where the Unix/Linux versions of the compiler could not
correctly digest Windows/DOS formatted files (with CR/LF).

iASL: Added a new option for "quiet mode" (-va) that produces only the
compilation summary, not individual errors and warnings. Useful for large
batch compilations.

AcpiExec: Implemented a new option (-z) to enable a forced semaphore/mutex
timeout that can be used to detect hang conditions during execution of AML
code (includes both internal semaphores and AML-defined mutexes and events.)
AcpiExec: Implemented a new option (-z) to enable a forced semaphore/mutex
timeout that can be used to detect hang conditions during execution of AML
code (includes both internal semaphores and AML-defined mutexes and events.)

Added new makefiles for the generation of acpica in a generic unix-like
environment. These makefiles are intended to generate the acpica tools and
environment. These makefiles are intended to generate the acpica tools and

utilities from the original acpica git source tree structure.

Test Suites: Updated and cleaned up the documentation files. Updated the copyrights to 2009, affecting all source files. Use the new version of iASL with quiet mode. Increased the number of available semaphores in the Windows OSL, allowing the aslts to execute fully on Windows. For the Unix OSL, added an alternate implementation of the semaphore timeout to allow aslts to execute fully on Cygwin.

---

20 March 2009. Summary of changes for version 20090320:

1) ACPI CA Core Subsystem:

Fixed a possible race condition between AcpiWalkNamespace and dynamic table unloads. Added a reader/writer locking mechanism to allow multiple concurrent namespace walks (readers), but block a dynamic table unload until it can gain exclusive write access to the namespace. This fixes a problem where a table unload could (possibly catastrophically) delete the portion of the namespace that is currently being examined by a walk. Adds a new file, utlock.c, that implements the reader/writer lock mechanism. ACPICA BZ 749.

Fixed a regression introduced in version 20090220 where a change to the FADT handling could cause the ACPICA subsystem to access non-existent I/O ports.

Modified the handling of FADT register and table (FACS/DSDT) addresses. The FADT can contain both 32-bit and 64-bit versions of these addresses. Previously, the 64-bit versions were favored, meaning that if both 32 and 64 versions were valid, but not equal, the 64-bit version was used. This was found to cause some machines to fail. Now, in this case, the 32-bit version is used instead. This now matches the Windows behavior.

Implemented a new mechanism to protect certain I/O ports. Provides Microsoft compatibility and protects the standard PC I/O ports from access via AML code. Adds a new file, hwvalid.c

Fixed a possible extraneous warning message from the FADT support. The message warns of a 32/64 length mismatch between the legacy and GAS definitions for a register.

Removed the obsolete AcpiOsValidateAddress OSL interface. This interface is made obsolete by the port protection mechanism above. It was previously used to validate the entire address range of an operation region, which could be incorrect if the range included illegal ports, but fields within the operation region did not actually access those ports. Validation is now performed on a per-field basis instead of the entire region.

Modified the handling of the PM1 Status Register ignored bit (bit 11.) Ignored bits must be "preserved" according to the ACPI spec. Usually, this means a read/modify/write when writing to the register. However, for status registers, writing a one means clear the event. Writing a zero means preserve the event (do not clear.) This behavior is clarified in the ACPI 4.0 spec, and the ACPICA code now simply always writes a zero to the ignored bit.

Modified the handling of ignored bits for the PM1 A/B Control Registers. As per the ACPI specification, for the control registers, preserve (read/modify/write) all bits that are defined as either reserved or ignored.

Updated the handling of write-only bits in the PM1 A/B Control Registers. When reading the register, zero the write-only bits as per the ACPI spec. ACPICA BZ 443. Lin Ming.

Removed "Linux" from the list of supported _OSI strings. Linux no longer wants to reply true to this request. The Windows strings are the only paths through the AML that are tested and known to work properly.

```
  Previous Release:
    Non-Debug Version:  82.0K Code, 17.5K Data,  99.5K Total
    Debug Version:     156.9K Code, 49.8K Data, 206.7K Total
  Current Release:
    Non-Debug Version:  82.6K Code, 17.6K Data, 100.2K Total
    Debug Version:     157.7K Code, 49.9K Data, 207.6K Total
```

2) iASL Compiler/Disassembler and Tools:

Acpiexec: Split the large aeexec.c file into two new files, aehandlers.c and aetables.c

---

20 February 2009. Summary of changes for version 20090220:

1) ACPI CA Core Subsystem:

Optimized the ACPI register locking. Removed locking for reads from the ACPI bit registers in PM1 Status, Enable, Control, and PM2 Control. The lock is
Optimized the ACPI register locking. Removed locking for reads from the ACPI bit registers in PM1 Status, Enable, Control, and PM2 Control. The lock is
not required when reading the single-bit registers. The **AcpiGetRegisterUnlocked function is no longer needed and has been removed. This will improve performance for reads on these registers. ACPICA BZ 760.**
AcpiGetRegisterUnlocked function is no longer needed and has been removed. This will improve performance for reads on these registers. ACPICA BZ 760.

Fixed the parameter validation for AcpiRead/Write. Now return **AE_BAD_PARAMETER if the input register pointer is null, and AE_BAD_ADDRESS if the register has an address of zero. Previously, these cases simply returned AE_OK. For optional registers such as PM1B status/enable/control, the caller**
AE_BAD_PARAMETER if the input register pointer is null, and AE_BAD_ADDRESS if the register has an address of zero. Previously, these cases simply returned AE_OK. For optional registers such as PM1B status/enable/control, the caller
should check for a valid register address before calling. ACPICA BZ 748.

Renamed the external ACPI bit register access functions. Renamed AcpiGetRegister and AcpiSetRegister to clarify the purpose of these **functions. The new names are AcpiReadBitRegister and AcpiWriteBitRegister. Also, restructured the code for these functions by simplifying the code path**
functions. The new names are AcpiReadBitRegister and AcpiWriteBitRegister. Also, restructured the code for these functions by simplifying the code path
and condensing duplicate code to reduce code size.

Added new functions to transparently handle the possibly split PM1 A/B **registers. AcpiHwReadMultiple and AcpiHwWriteMultiple. These two functions now handle the split registers for PM1 Status, Enable, and Control. ACPICA BZ**
registers. AcpiHwReadMultiple and AcpiHwWriteMultiple. These two functions now handle the split registers for PM1 Status, Enable, and Control. ACPICA BZ
746.

Added a function to handle the PM1 control registers, AcpiHwWritePm1Control. This function writes both of the PM1 control registers (A/B). These registers are different than the PM1 A/B status and enable registers in that different values can be written to the A/B registers. Most notably, the SLP_TYP bits can be different, as per the values returned from the _Sx predefined methods.
Added a function to handle the PM1 control registers, AcpiHwWritePm1Control. This function writes both of the PM1 control registers (A/B). These registers are different than the PM1 A/B status and enable registers in that different values can be written to the A/B registers. Most notably, the SLP_TYP bits can be different, as per the values returned from the _Sx predefined methods.

Removed an extra register write within AcpiHwClearAcpiStatus. This function was writing an optional PM1B status register twice. The existing call to the low-level AcpiHwRegisterWrite automatically handles a possibly split PM1 A/B
Removed an extra register write within AcpiHwClearAcpiStatus. This function was writing an optional PM1B status register twice. The existing call to the low-level AcpiHwRegisterWrite automatically handles a possibly split PM1 A/B
register. ACPICA BZ 751.

Split out the PM1 Status registers from the FADT. Added new globals for these
Split out the PM1 Status registers from the FADT. Added new globals for these
registers (A/B), similar to the way the PM1 Enable registers are handled. **Instead of overloading the FADT Event Register blocks. This makes the code**
Instead of overloading the FADT Event Register blocks. This makes the code
clearer and less prone to error.

Fixed the warning message for when the platform contains too many ACPI tables for the default size of the global root table data structure. The calculation
Fixed the warning message for when the platform contains too many ACPI tables for the default size of the global root table data structure. The calculation
for the truncation value was incorrect.

Removed the ACPI_GET_OBJECT_TYPE macro. Removed all instances of this **obsolete macro, since it is now a simple reference to ->common.type. There**
obsolete macro, since it is now a simple reference to ->common.type. There
were about 150 invocations of the macro across 41 files. ACPICA BZ 755.

Removed the redundant ACPI_BITREG_SLEEP_TYPE_B. This type is the same as TYPE_A. Removed this and all related instances. Renamed SLEEP_TYPE_A to simply SLEEP_TYPE. ACPICA BZ 754.

Conditionally compile the AcpiSetFirmwareWakingVector64 function. This function is only needed on 64-bit host operating systems and is thus not included for 32-bit hosts.

Debug output: print the input and result for invocations of the _OSI reserved control method via the ACPI_LV_INFO debug level. Also, reduced some of the
Debug output: print the input and result for invocations of the _OSI reserved control method via the ACPI_LV_INFO debug level. Also, reduced some of the
verbosity of this debug level. Len Brown.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  82.3K Code, 17.5K Data,  99.8K Total
    Debug Version:     157.3K Code, 49.8K Data, 207.1K Total
  Current Release:
    Non-Debug Version:  82.0K Code, 17.5K Data,  99.5K Total
    Debug Version:     156.9K Code, 49.8K Data, 206.7K Total
```

2) iASL Compiler/Disassembler and Tools:

Disassembler: Decode the FADT PM_Profile field. Emit ascii names for the
various legal performance profiles.

---

23 January 2009. Summary of changes for version 20090123:

1) ACPI CA Core Subsystem:

Added the 2009 copyright to all module headers and signons. This affects
virtually every file in the ACPICA core subsystem, the iASL compiler, and
the tools/utilities.

**Implemented a change to allow the host to override any ACPI table, including
dynamically loaded tables. Previously, only the DSDT could be replaced by the
host. With this change, the AcpiOsTableOverride interface is called for each
table found in the RSDT/XSDT during ACPICA initialization, and also whenever**
*Implemented a change to allow the host to override any ACPI table, including
dynamically loaded tables. Previously, only the DSDT could be replaced by the
host. With this change, the AcpiOsTableOverride interface is called for each
table found in the RSDT/XSDT during ACPICA initialization, and also whenever*
a table is dynamically loaded via the AML Load operator.

Updated FADT flag definitions, especially the Boot Architecture flags.

**Debugger: For the Find command, automatically pad the input ACPI name with
underscores if the name is shorter than 4 characters. This enables a match**
*Debugger: For the Find command, automatically pad the input ACPI name with
underscores if the name is shorter than 4 characters. This enables a match*
with the actual namespace entry which is itself padded with underscores.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  82.3K Code, 17.4K Data,  99.7K Total
    Debug Version:     157.1K Code, 49.7K Data, 206.8K Total
  Current Release:
    Non-Debug Version:  82.3K Code, 17.5K Data,  99.8K Total
    Debug Version:     157.3K Code, 49.8K Data, 207.1K Total
```

2) iASL Compiler/Disassembler and Tools:

Fix build error under Bison-2.4.

Dissasembler: Enhanced FADT support. Added decoding of the Boot Architecture
Dissasembler: Enhanced FADT support. Added decoding of the Boot Architecture
flags. Now decode all flags, regardless of the FADT version. Flag output
includes the FADT version which first defined each flag.

The iASL -g option now dumps the RSDT to a file (in addition to the FADT and
The iASL -g option now dumps the RSDT to a file (in addition to the FADT and
DSDT). Windows only.

---

04 December 2008. Summary of changes for version 20081204:

1) ACPI CA Core Subsystem:

The ACPICA Programmer Reference has been completely updated and revamped for
The ACPICA Programmer Reference has been completely updated and revamped for
this release. This includes updates to the external interfaces, OSL
interfaces, the overview sections, and the debugger reference.

Several new ACPICA interfaces have been implemented and documented in the
programmer reference:
AcpiReset - Writes the reset value to the FADT-defined reset register.
AcpiDisableAllGpes - Disable all available GPEs.
AcpiEnableAllRuntimeGpes - Enable all available runtime GPEs.
AcpiGetGpeDevice - Get the GPE block device associated with a GPE.
AcpiGbl_CurrentGpeCount - Tracks the current number of available GPEs.
AcpiRead - Low-level read ACPI register (was HwLowLevelRead.)
AcpiWrite - Low-level write ACPI register (was HwLowLevelWrite.)

Most of the public ACPI hardware-related interfaces have been moved to a new
Most of the public ACPI hardware-related interfaces have been moved to a new
file, components/hardware/hwxface.c

Enhanced the FADT parsing and low-level ACPI register access: The ACPI
register lengths within the FADT are now used, and the low level ACPI
register access no longer hardcodes the ACPI register lengths. Given that
**there may be some risk in actually trusting the FADT register lengths, a run-
time option was added to fall back to the default hardcoded lengths if the**
*there may be some risk in actually trusting the FADT register lengths, a run-
time option was added to fall back to the default hardcoded lengths if the*
FADT proves to contain incorrect values - UseDefaultRegisterWidths. This
option is set to true for now, and a warning is issued if a suspicious FADT
option is set to true for now, and a warning is issued if a suspicious FADT
register length is overridden with the default value.

Fixed a reference count issue in NsRepairObject. This problem was introduced
Fixed a reference count issue in NsRepairObject. This problem was introduced
in version 20081031 as part of a fix to repair Buffer objects within
Packages. Lin Ming.

Added semaphore support to the Linux/Unix application OS-services layer
(OSL). ACPICA BZ 448. Lin Ming.

Added the ACPI_MUTEX_TYPE configuration option to select whether mutexes will
Added the ACPI_MUTEX_TYPE configuration option to select whether mutexes will
be implemented in the OSL, or will binary semaphores be used instead.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  81.7K Code, 17.3K Data,  99.0K Total
    Debug Version:     156.4K Code, 49.4K Data, 205.8K Total
  Current Release:
    Non-Debug Version:  82.3K Code, 17.4K Data,  99.7K Total
```

Debug Version: 157.1K Code, 49.7K Data, 206.8K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Completed the '-e' option to include additional ACPI tables in order to aid with disassembly and External statement generation. ACPICA BZ 742. Lin
iASL: Completed the '-e' option to include additional ACPI tables in order to aid with disassembly and External statement generation. ACPICA BZ 742. Lin Ming.

iASL: Removed the "named object in while loop" error. The compiler cannot determine how many times a loop will execute. ACPICA BZ 730.

Disassembler: Implemented support for FADT revision 2 (MS extension). ACPICA
Disassembler: Implemented support for FADT revision 2 (MS extension). ACPICA BZ 743.

Disassembler: Updates for several ACPI data tables (HEST, EINJ, and MCFG).
Disassembler: Updates for several ACPI data tables (HEST, EINJ, and MCFG).

---
31 October 2008. Summary of changes for version 20081031:

1) ACPI CA Core Subsystem:

Restructured the ACPICA header files into public/private. acpi.h now includes only the "public" acpica headers. All other acpica headers are "private" and should not be included by acpica users. One new file, accommon.h is used to include the commonly used private headers for acpica code generation. Future
Restructured the ACPICA header files into public/private. acpi.h now includes only the "public" acpica headers. All other acpica headers are "private" and should not be included by acpica users. One new file, accommon.h is used to include the commonly used private headers for acpica code generation. Future plans include moving all private headers to a new subdirectory.

Implemented an automatic Buffer->String return value conversion for **predefined ACPI methods. For these methods (such as _BIF), added automatic conversion for return objects that are required to be a String, but a Buffer was found instead. This can happen when reading string battery data from an operation region, because it used to be difficult to convert the data from buffer to string from within the ASL. Ensures that the host OS is provided**
predefined ACPI methods. For these methods (such as _BIF), added automatic conversion for return objects that are required to be a String, but a Buffer was found instead. This can happen when reading string battery data from an operation region, because it used to be difficult to convert the data from buffer to string from within the ASL. Ensures that the host OS is provided with a valid null-terminated string. Linux BZ 11822.

Updated the FACS waking vector interfaces. Split AcpiSetFirmwareWakingVector into two: one for the 32-bit vector, another for the 64-bit vector. This is

required because the host OS must setup the wake much differently for each vector (real vs. protected mode, etc.) and the interface itself should not be deciding which vector to use. Also, eliminated the GetFirmwareWakingVector interface, as it served no purpose (only the firmware reads the vector, OS
Updated the FACS waking vector interfaces. Split AcpiSetFirmwareWakingVector into two: one for the 32-bit vector, another for the 64-bit vector. This is required because the host OS must setup the wake much differently for each vector (real vs. protected mode, etc.) and the interface itself should not be deciding which vector to use. Also, eliminated the GetFirmwareWakingVector interface, as it served no purpose (only the firmware reads the vector, OS only writes the vector.) ACPICA BZ 731.

Implemented a mechanism to escape infinite AML While() loops. Added a loop counter to force exit from AML While loops if the count becomes too large.
Implemented a mechanism to escape infinite AML While() loops. Added a loop counter to force exit from AML While loops if the count becomes too large. This can occur in poorly written AML when the hardware does not respond **within a while loop and the loop does not implement a timeout. The maximum loop count is configurable. A new exception code is returned when a loop is**
within a while loop and the loop does not implement a timeout. The maximum loop count is configurable. A new exception code is returned when a loop is broken, AE_AML_INFINITE_LOOP. Alexey Starikovskiy, Bob Moore.

Optimized the execution of AML While loops. Previously, a control state object was allocated and freed for each execution of the loop. The **optimization is to simply reuse the control state for each iteration. This**
optimization is to simply reuse the control state for each iteration. This speeds up the raw loop execution time by about 5%.

Enhanced the implicit return mechanism. For Windows compatibility, return an implicit integer of value zero for methods that contain no executable code.
Enhanced the implicit return mechanism. For Windows compatibility, return an implicit integer of value zero for methods that contain no executable code. Such methods are seen in the field as stubs (presumably), and can cause drivers to fail if they expect a return value. Lin Ming.

Allow multiple backslashes as root prefixes in namepaths. In a fully **qualified namepath, allow multiple backslash prefixes. This can happen (and is seen in the field) because of the use of a double-backslash in strings (since backslash is the escape character) causing confusion. ACPICA BZ 739**
qualified namepath, allow multiple backslash prefixes. This can happen (and is seen in the field) because of the use of a double-backslash in strings *(since backslash is the escape character) causing confusion. ACPICA BZ 739* Lin Ming.

Emit a warning if two different FACS or DSDT tables are discovered in the **FADT. Checks if there are two valid but different addresses for the FACS and**
FADT. Checks if there are two valid but different addresses for the FACS and DSDT within the FADT (mismatch between the 32-bit and 64-bit fields.)

Consolidated the method argument count validation code. Merged the code that

Consolidated the method argument count validation code. Merged the code that validates control method argument counts into the predefined validation module. Eliminates possible multiple warnings for incorrect argument counts.

Implemented ACPICA example code. Includes code for ACPICA initialization, handler installation, and calling a control method. Available at source/tools/examples.

Added a global pointer for FACS table to simplify internal FACS access. Use the global pointer instead of using AcpiGetTableByIndex for each FACS access. This simplifies the code for the Global Lock and the Firmware Waking Vector(s).

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

  Previous Release:
    Non-Debug Version: 81.2K Code, 17.0K Data, 98.2K Total
    Debug Version: 155.8K Code, 49.1K Data, 204.9K Total
  Current Release:
    Non-Debug Version: 81.7K Code, 17.3K Data, 99.0K Total
    Debug Version: 156.4K Code, 49.4K Data, 205.8K Total

2) iASL Compiler/Disassembler and Tools:

iASL: Improved disassembly of external method calls. Added the -e option to allow the inclusion of additional ACPI tables to help with the disassembly of method invocations and the generation of external declarations during the disassembly. Certain external method invocations cannot be disassembled properly without the actual declaration of the method. Use the -e option to include the table where the external method(s) are actually declared. Most useful for disassembling SSDTs that make method calls back to the master DSDT. Lin Ming. Example: To disassemble an SSDT with calls to DSDT: iasl -d -e dsdt.aml ssdt1.aml

iASL: Fix to allow references to aliases within ASL namepaths. Fixes a problem where the use of an alias within a namepath would result in a not found error or cause the compiler to fault. Also now allows forward references from the Alias operator itself. ACPICA BZ 738.

---

26 September 2008. Summary of changes for version 20080926:

1) ACPI CA Core Subsystem:

Designed and implemented a mechanism to validate predefined ACPI methods and objects. This code validates the predefined ACPI objects (objects whose names start with underscore) that appear in the namespace, at the time they are evaluated. The argument count and the type of the returned object are validated against the ACPI specification. The purpose of this validation is to detect problems with the BIOS-implemented predefined ACPI objects before the results are returned to the ACPI-related drivers. Future enhancements may include actual repair of incorrect return objects where possible. Two new files are nspredef.c and acpredef.h.

Fixed a fault in the AML parser if a memory allocation fails during the Op completion routine AcpiPsCompleteThisOp. Lin Ming. ACPICA BZ 492.

Fixed an issue with implicit return compatibility. This change improves the implicit return mechanism to be more compatible with the MS interpreter. Lin Ming, ACPICA BZ 349.

Implemented support for zero-length buffer-to-string conversions. Allow zero length strings during interpreter buffer-to-string conversions. For example, during the ToDecimalString and ToHexString operators, as well as implicit conversions. Fiodor Suietov, ACPICA BZ 585.

Fixed two possible memory leaks in the error exit paths of AcpiUtUpdateObjectReference and AcpiUtWalkPackageTree. These functions are similar in that they use a stack of state objects in order to eliminate recursion. The stack must be fully unwound and deallocated if an error occurs. Lin Ming. ACPICA BZ 383.

Removed the unused ACPI_BITREG_WAKE_ENABLE definition and entry in the global ACPI register table. This bit does not exist and is unused. Lin Ming, Bob Moore ACPICA BZ 442.

Removed the obsolete version number in module headers. Removed the "$Revision" number that appeared in each module header. This version number was useful under SourceSafe and CVS, but has no meaning under git. It is not only incorrect, it could also be misleading.

Example Code and Data Size: These are the sizes for the OS-independent

acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.7K Code, 16.4K Data,  96.1K Total
    Debug Version:     153.7K Code, 48.2K Data, 201.9K Total
  Current Release:
    Non-Debug Version:  81.2K Code, 17.0K Data,  98.2K Total
    Debug Version:     155.8K Code, 49.1K Data, 204.9K Total
```

---

29 August 2008. Summary of changes for version 20080829:

1) ACPI CA Core Subsystem:

Completed a major cleanup of the internal ACPI_OPERAND_OBJECT of type
Reference. Changes include the elimination of cheating on the Object field
Reference. Changes include the elimination of cheating on the Object field
for the DdbHandle subtype, addition of a reference class field to
differentiate the various reference types (instead of an AML opcode), and the
differentiate the various reference types (instead of an AML opcode), and the
cleanup of debug output for this object. Lin Ming, Bob Moore. BZ 723

Reduce an error to a warning for an incorrect method argument count.
Previously aborted with an error if too few arguments were passed to a
control method via the external ACPICA interface. Now issue a warning instead and continue. Handles the case where the method inadvertently declares too many arguments, but does not actually use the extra ones. Applies mainly to
control method via the external ACPICA interface. Now issue a warning instead and continue. Handles the case where the method inadvertently declares too many arguments, but does not actually use the extra ones. Applies mainly to
the predefined methods. Lin Ming. Linux BZ 11032.

Disallow the evaluation of named object types with no intrinsic value. Return AE_TYPE for objects that have no value and therefore evaluation is undefined: Device, Event, Mutex, Region, Thermal, and Scope. Previously, evaluation of these types were allowed, but an exception would be generated at some point
Disallow the evaluation of named object types with no intrinsic value. Return AE_TYPE for objects that have no value and therefore evaluation is undefined: Device, Event, Mutex, Region, Thermal, and Scope. Previously, evaluation of these types were allowed, but an exception would be generated at some point
during the evaluation. Now, the error is generated up front.

Fixed a possible memory leak in the AcpiNsGetExternalPathname function
(nsnames.c). Fixes a leak in the error exit path.

Removed the obsolete debug levels ACPI_DB_WARN and ACPI_DB_ERROR. These debug levels were made obsolete by the ACPI_WARNING, ACPI_ERROR, and ACPI_EXCEPTION
Removed the obsolete debug levels ACPI_DB_WARN and ACPI_DB_ERROR. These debug levels were made obsolete by the ACPI_WARNING, ACPI_ERROR, and ACPI_EXCEPTION
interfaces. Also added ACPI_DB_EVENTS to correspond with the existing
ACPI_LV_EVENTS.

Removed obsolete and/or unused exception codes from the acexcep.h header.
There is the possibility that certain device drivers may be affected if they
There is the possibility that certain device drivers may be affected if they
use any of these exceptions.

The ACPICA documentation has been added to the public git source tree, under
The ACPICA documentation has been added to the public git source tree, under
acpica/documents. Included are the ACPICA programmer reference, the iASL
compiler reference, and the changes.txt release logfile.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.7K Code, 16.4K Data,  96.1K Total
    Debug Version:     153.9K Code, 48.4K Data, 202.3K Total
  Current Release:
    Non-Debug Version:  79.7K Code, 16.4K Data,  96.1K Total
    Debug Version:     153.7K Code, 48.2K Data, 201.9K Total
```

2) iASL Compiler/Disassembler and Tools:

Allow multiple argument counts for the predefined _SCP method. ACPI 3.0
defines _SCP with 3 arguments. Previous versions defined it with only 1
argument. iASL now allows both definitions.

iASL/disassembler: avoid infinite loop on bad ACPI tables. Check for zero-length subtables when disassembling ACPI tables. Also fixed a couple of errors where a full 16-bit table type field was not extracted from the input
iASL/disassembler: avoid infinite loop on bad ACPI tables. Check for zero-length subtables when disassembling ACPI tables. Also fixed a couple of errors where a full 16-bit table type field was not extracted from the input
properly.

acpisrc: Improve comment counting mechanism for generating source code
statistics. Count first and last lines of multi-line comments as whitespace, not comment lines. Handle Linux legal header in addition to standard acpica
statistics. Count first and last lines of multi-line comments as whitespace, not comment lines. Handle Linux legal header in addition to standard acpica
header.

---

29 July 2008. Summary of changes for version 20080729:

1) ACPI CA Core Subsystem:

Fix a possible deadlock in the GPE dispatch. Remove call to
AcpiHwDisableAllGpes during wake in AcpiEvGpeDispatch. This call will attempt to acquire the GPE lock but can deadlock since the GPE lock is already held at dispatch time. This code was introduced in version 20060831 as a response
AcpiHwDisableAllGpes during wake in AcpiEvGpeDispatch. This call will attempt to acquire the GPE lock but can deadlock since the GPE lock is already held

at dispatch time. This code was introduced in version 20060831 as a response
to Linux BZ 6881 and has since been removed from Linux.

Add a function to dereference returned reference objects. Examines the return
object from a call to AcpiEvaluateObject. Any Index or RefOf references are
automatically dereferenced in an attempt to return something useful (these
reference types cannot be converted into an external ACPI_OBJECT.) Provides
Add a function to dereference returned reference objects. Examines the return
object from a call to AcpiEvaluateObject. Any Index or RefOf references are
automatically dereferenced in an attempt to return something useful (these
reference types cannot be converted into an external ACPI_OBJECT.) Provides
MS compatibility. Lin Ming, Bob Moore. Linux BZ 11105

x2APIC support: changes for MADT and SRAT ACPI tables. There are 2 new
subtables for the MADT and one new subtable for the SRAT. Includes
disassembler and AcpiSrc support. Data from the Intel 64 Architecture x2APIC
disassembler and AcpiSrc support. Data from the Intel 64 Architecture x2APIC
Specification, June 2008.

Additional error checking for pathname utilities. Add error check after all
Additional error checking for pathname utilities. Add error check after all
calls to AcpiNsGetPathnameLength. Add status return from
AcpiNsBuildExternalPath and check after all calls. Add parameter validation
AcpiNsBuildExternalPath and check after all calls. Add parameter validation
to AcpiUtInitializeBuffer. Reported by and initial patch by Ingo Molnar.

Return status from the global init function AcpiUtGlobalInitialize. This is
used by both the kernel subsystem and the utilities such as iASL compiler.
The function could possibly fail when the caches are initialized. Yang Yi.
Return status from the global init function AcpiUtGlobalInitialize. This is
used by both the kernel subsystem and the utilities such as iASL compiler.
The function could possibly fail when the caches are initialized. Yang Yi.

Add a function to decode reference object types to strings. Created for
improved error messages.

Improve object conversion error messages. Better error messages during object
conversion from internal to the external ACPI_OBJECT. Used for external calls
Improve object conversion error messages. Better error messages during object
conversion from internal to the external ACPI_OBJECT. Used for external calls
to AcpiEvaluateObject.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.6K Code, 16.2K Data,  95.8K Total
    Debug Version:     153.5K Code, 48.2K Data, 201.7K Total
  Current Release:
    Non-Debug Version:  79.7K Code, 16.4K Data,  96.1K Total
```



```
    Debug Version:     153.9K Code, 48.4K Data, 202.3K Total
```

2) iASL Compiler/Disassembler and Tools:

Debugger: fix a possible hang when evaluating non-methods. Fixes a problem
introduced in version 20080701. If the object being evaluated (via execute
command) is not a method, the debugger can hang while trying to obtain non-
Debugger: fix a possible hang when evaluating non-methods. Fixes a problem
introduced in version 20080701. If the object being evaluated (via execute
command) is not a method, the debugger can hang while trying to obtain non-
existent parameters.

iASL: relax error for using reserved "_T_x" identifiers. These names can
appear in a disassembled ASL file if they were emitted by the original
compiler. Instead of issuing an error or warning and forcing the user to
manually change these names, issue a remark instead.

iASL: error if named object created in while loop. Emit an error if any named
object is created within a While loop. If allowed, this code will generate a
run-time error on the second iteration of the loop when an attempt is made to
iASL: error if named object created in while loop. Emit an error if any named
object is created within a While loop. If allowed, this code will generate a
run-time error on the second iteration of the loop when an attempt is made to
create the same named object twice. ACPICA bugzilla 730.

iASL: Support absolute pathnames for include files. Add support for absolute
pathnames within the Include operator. previously, only relative pathnames
iASL: Support absolute pathnames for include files. Add support for absolute
pathnames within the Include operator. previously, only relative pathnames
were supported.

iASL: Enforce minimum 1 interrupt in interrupt macro and Resource Descriptor.
iASL: Enforce minimum 1 interrupt in interrupt macro and Resource Descriptor.
The ACPI spec requires one interrupt minimum. BZ 423

iASL: Handle a missing ResourceSource arg, with a present SourceIndex.
Handles the case for the Interrupt Resource Descriptor where
the ResourceSource argument is omitted but ResourceSourceIndex
is present. Now leave room for the Index. BZ 426

iASL: Prevent error message if CondRefOf target does not exist. Fixes cases
iASL: Prevent error message if CondRefOf target does not exist. Fixes cases
where an error message is emitted if the target does not exist. BZ 516

iASL: Fix broken -g option (get Windows ACPI tables). Fixes the -g option
(get ACPI tables on Windows). This was apparently broken in version 20070919.
(get ACPI tables on Windows). This was apparently broken in version 20070919.

AcpiXtract: Handle EOF while extracting data. Correctly handle the case where
AcpiXtract: Handle EOF while extracting data. Correctly handle the case where
the EOF happens immediately after the last table in the input file. Print
completion message. Previously, no message was displayed in this case.

```
5219 ----------------------------------------
5220 01 July 2008. Summary of changes for version 20080701:

5222 0) Git source tree / acpica.org

5224 Fixed a problem where a git-clone from http would not transfer the entire
5225 source tree.

5227 1) ACPI CA Core Subsystem:

5229 Implemented a "careful" GPE disable in AcpiEvDisableGpe, only modify one
5230 enable bit. Now performs a read-change-write of the enable register
5231 instead
5232 of simply writing out the cached enable mask. This will prevent
5233 inadvertent
5234 enabling of GPEs if a rogue GPE is received during initialization (before
5235 GPE
2145 enable bit. Now performs a read-change-write of the enable register instead
2146 of simply writing out the cached enable mask. This will prevent inadvertent
2147 enabling of GPEs if a rogue GPE is received during initialization (before GPE
5236 handlers are installed.)

5238 Implemented a copy for dynamically loaded tables. Previously, dynamically
5239 loaded tables were simply mapped - but on some machines this memory is
5240 corrupted after suspend. Now copy the table to a local buffer. For the
5241 OpRegion case, added checksum verify. Use the table length from the table
5242 header, not the region length. For the Buffer case, use the table length
5243 also. Dennis Noordsij, Bob Moore. BZ 10734

5245 Fixed a problem where the same ACPI table could not be dynamically loaded
5246 and
5247 unloaded more than once. Without this change, a table cannot be loaded
5248 again
2157 Fixed a problem where the same ACPI table could not be dynamically loaded and
2158 unloaded more than once. Without this change, a table cannot be loaded again
5249 once it has been loaded/unloaded one time. The current mechanism does not
5250 unregister a table upon an unload. During a load, if the same table is
5251 found,
2160 unregister a table upon an unload. During a load, if the same table is found,
5252 this no longer returns an exception. BZ 722

5254 Fixed a problem where the wrong descriptor length was calculated for the
5255 EndTag descriptor in 64-bit mode. The "minimal" descriptors such as
5256 EndTag
2164 EndTag descriptor in 64-bit mode. The "minimal" descriptors such as EndTag
5257 are calculated as 12 bytes long, but the actual length in the internal
5258 descriptor is 16 because of the round-up to 8 on the 64-bit build.
5259 Reported
2166 descriptor is 16 because of the round-up to 8 on the 64-bit build. Reported
5260 by Linn Crosetto. BZ 728

5262 Fixed a possible memory leak in the Unload operator. The DdbHandle
5263 returned
5264 by Load() did not have its reference count decremented during unload,
5265 leading
2169 Fixed a possible memory leak in the Unload operator. The DdbHandle returned
2170 by Load() did not have its reference count decremented during unload, leading
5266 to a memory leak. Lin Ming. BZ 727

5268 Fixed a possible memory leak when deleting thermal/processor objects. Any
5269 associated notify handlers (and objects) were not being deleted. Fiodor
5270 Suietov. BZ 506

5272 Fixed the ordering of the ASCII names in the global mutex table to match
5273 the
```



```
5274 actual mutex IDs. Used by AcpiUtGetMutexName, a function used for debug
5275 only.
2177 Fixed the ordering of the ASCII names in the global mutex table to match the
2178 actual mutex IDs. Used by AcpiUtGetMutexName, a function used for debug only.
5276 Vegard Nossum. BZ 726

5278 Enhanced the AcpiGetObjectInfo interface to return the number of required
5279 arguments if the object is a control method. Added this call to the
5280 debugger
2182 arguments if the object is a control method. Added this call to the debugger
5281 so the proper number of default arguments are passed to a method. This
5282 prevents a warning when executing methods from AcpiExec.

5284 Added a check for an invalid handle in AcpiGetObjectInfo. Return
5285 AE_BAD_PARAMETER if input handle is invalid. BZ 474

5287 Fixed an extraneous warning from exconfig.c on the 64-bit build.

5289 Example Code and Data Size: These are the sizes for the OS-independent
5290 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
5291 debug version of the code includes the debug output trace mechanism and
5292 has a
2193 debug version of the code includes the debug output trace mechanism and has a
5293 much larger code and data size.

5295   Previous Release:
5296     Non-Debug Version:  79.3K Code, 16.2K Data,  95.5K Total
5297     Debug Version:     153.0K Code, 48.2K Data, 201.2K Total
5298   Current Release:
5299     Non-Debug Version:  79.6K Code, 16.2K Data,  95.8K Total
5300     Debug Version:     153.5K Code, 48.2K Data, 201.7K Total

5302 2) iASL Compiler/Disassembler and Tools:

5304 iASL: Added two missing ACPI reserved names. Added _MTP and _ASZ, both
5305 resource descriptor names.

5307 iASL: Detect invalid ASCII characters in input (windows version). Removed
5308 the
2208 iASL: Detect invalid ASCII characters in input (windows version). Removed the
5309 "-CF" flag from the flex compile, enables correct detection of non-ASCII
5310 characters in the input. BZ 441

5312 iASL: Eliminate warning when result of LoadTable is not used. Eliminate
5313 the
2212 iASL: Eliminate warning when result of LoadTable is not used. Eliminate the
5314 "result of operation not used" warning when the DDB handle returned from
5315 LoadTable is not used. The warning is not needed. BZ 590

5317 AcpiExec: Add support for dynamic table load/unload. Now calls _CFG
5318 method
5319 to
5320 pass address of table to the AML. Added option to disable OpRegion
5321 simulation
5322 to allow creation of an OpRegion with a real address that was passed to
5323 _CFG.
5324 All of this allows testing of the Load and Unload operators from
5325 AcpiExec.
2216 AcpiExec: Add support for dynamic table load/unload. Now calls _CFG method to
2217 pass address of table to the AML. Added option to disable OpRegion simulation
2218 to allow creation of an OpRegion with a real address that was passed to _CFG.
2219 All of this allows testing of the Load and Unload operators from AcpiExec.

5327 Debugger: update tables command for unloaded tables. Handle unloaded
5328 tables
2221 Debugger: update tables command for unloaded tables. Handle unloaded tables
```

and use the standard table header output routine.

---
09 June 2008. Summary of changes for version 20080609:

1) ACPI CA Core Subsystem:

Implemented a workaround for reversed _PRT entries. A significant number of
Implemented a workaround for reversed _PRT entries. A significant number of
BIOSs erroneously reverse the _PRT SourceName and the SourceIndex. This
change dynamically detects and repairs this problem. Provides compatibility
change dynamically detects and repairs this problem. Provides compatibility
with MS ACPI. BZ 6859

Simplified the internal ACPI hardware interfaces to eliminate the locking
flag parameter from Register Read/Write. Added a new external interface,
AcpiGetRegisterUnlocked.

Fixed a problem where the invocation of a GPE control method could hang. This
Fixed a problem where the invocation of a GPE control method could hang. This
was a regression introduced in 20080514. The new method argument count
validation mechanism can enter an infinite loop when a GPE method is
dispatched. Problem fixed by removing the obsolete code that passed GPE block
information to the notify handler via the control method parameter pointer.
dispatched. Problem fixed by removing the obsolete code that passed GPE block
information to the notify handler via the control method parameter pointer.

Fixed a problem where the _SST execution status was incorrectly returned to
Fixed a problem where the _SST execution status was incorrectly returned to
the caller of AcpiEnterSleepStatePrep. This was a regression introduced in
20080514. _SST is optional and a NOT_FOUND exception should never be
returned. BZ 716

Fixed a problem where a deleted object could be accessed from within the AML
Fixed a problem where a deleted object could be accessed from within the AML
parser. This was a regression introduced in version 20080123 as a fix for the
Unload operator. Lin Ming. BZ 10669

Cleaned up the debug operand dump mechanism. Eliminated unnecessary operands
Cleaned up the debug operand dump mechanism. Eliminated unnecessary operands
and eliminated the use of a negative index in a loop. Operands are now
displayed in the correct order, not backwards. This also fixes a regression
displayed in the correct order, not backwards. This also fixes a regression
introduced in 20080514 on 64-bit systems where the elimination of
ACPI_NATIVE_UINT caused the negative index to go large and positive. BZ 715
ACPI_NATIVE_UINT caused the negative index to go large and positive. BZ 715

Fixed a possible memory leak in EvPciConfigRegionSetup where the error exit
Fixed a possible memory leak in EvPciConfigRegionSetup where the error exit
path did not delete a locally allocated structure.

Updated definitions for the DMAR and SRAT tables to synchronize with the
current specifications. Includes disassembler support.

Fixed a problem in the mutex debug code (in utmutex.c) where an incorrect
loop termination value was used. Loop terminated on iteration early, missing
loop termination value was used. Loop terminated on iteration early, missing
one mutex. Linn Crosetto

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has a
debug version of the code includes the debug output trace mechanism and has a
much larger code and data size.

  Previous Release:
    Non-Debug Version: 79.5K Code, 16.2K Data, 95.7K Total
    Debug Version: 153.3K Code, 48.3K Data, 201.6K Total
  Current Release:
    Non-Debug Version: 79.3K Code, 16.2K Data, 95.5K Total
    Debug Version: 153.0K Code, 48.2K Data, 201.2K Total

2) iASL Compiler/Disassembler and Tools:

Disassembler: Implemented support for EisaId() within _CID objects. Now
disassemble integer _CID objects back to EisaId invocations, including
multiple integers within _CID packages. Includes single-step support for
debugger also.

Disassembler: Added support for DMAR and SRAT table definition changes.

---
14 May 2008. Summary of changes for version 20080514:

1) ACPI CA Core Subsystem:

Fixed a problem where GPEs were enabled too early during the ACPICA
initialization. This could lead to "handler not installed" errors on some
machines. Moved GPE enable until after _REG/_STA/_INI methods are run. This
machines. Moved GPE enable until after _REG/_STA/_INI methods are run. This
ensures that all operation regions and devices throughout the namespace have
ensures that all operation regions and devices throughout the namespace have
been initialized before GPEs are enabled. Alexey Starikovskiy, BZ 9916.

Implemented a change to the enter sleep code. Moved execution of the _GTS
method to just before setting sleep enable bit. The execution was moved from
method to just before setting sleep enable bit. The execution was moved from
AcpiEnterSleepStatePrep to AcpiEnterSleepState. _GTS is now executed
immediately before the SLP_EN bit is set, as per the ACPI specification.
Luming Yu, BZ 1653.

Implemented a fix to disable unknown GPEs (2nd version). Now always disable
Implemented a fix to disable unknown GPEs (2nd version). Now always disable
the GPE, even if ACPICA thinks that that it is already disabled. It is
possible that the AML or some other code has enabled the GPE unbeknownst to
possible that the AML or some other code has enabled the GPE unbeknownst to
the ACPICA code.

Fixed a problem with the Field operator where zero-length fields would return

an AE_AML_NO_OPERAND exception during table load. Fix enables zero-length ASL
Fixed a problem with the Field operator where zero-length fields would return an AE_AML_NO_OPERAND exception during table load. Fix enables zero-length ASL field declarations in Field(), BankField(), and IndexField(). BZ 10606.

Implemented a fix for the Load operator, now load the table at the namespace root. This reverts a change introduced in version 20071019. The table is now
Implemented a fix for the Load operator, now load the table at the namespace root. This reverts a change introduced in version 20071019. The table is now loaded at the namespace root even though this goes against the ACPI **specification. This provides compatibility with other ACPI implementations. The ACPI specification will be updated to reflect this in ACPI 4.0. Lin Ming.**
specification. This provides compatibility with other ACPI implementations. The ACPI specification will be updated to reflect this in ACPI 4.0. Lin Ming.

Fixed a problem where ACPICA would not Load() tables with unusual signatures.
Fixed a problem where ACPICA would not Load() tables with unusual signatures. Now ignore ACPI table signature for Load() operator. Only "SSDT" is acceptable to the ACPI spec, but tables are seen with OEMx and null sigs. **Therefore, signature validation is worthless. Apparently MS ACPI accepts such**
Therefore, signature validation is worthless. Apparently MS ACPI accepts such signatures, ACPICA must be compatible. BZ 10454.

Fixed a possible negative array index in AcpiUtValidateException. Added NULL fields to the exception string arrays to eliminate a -1 subtraction on the
Fixed a possible negative array index in AcpiUtValidateException. Added NULL fields to the exception string arrays to eliminate a -1 subtraction on the SubStatus field.

Updated the debug tracking macros to reduce overall code and data size. Changed ACPI_MODULE_NAME and ACPI_FUNCTION_NAME to use arrays of strings instead of pointers to static strings. Jan Beulich and Bob Moore.

Implemented argument count checking in control method invocation via **AcpiEvaluateObject. Now emit an error if too few arguments, warning if too many. This applies only to extern programmatic control method execution, not**
AcpiEvaluateObject. Now emit an error if too few arguments, warning if too many. This applies only to extern programmatic control method execution, not method-to-method calls within the AML. Lin Ming.

Eliminated the ACPI_NATIVE_UINT type across all ACPICA code. This type is no longer needed, especially with the removal of 16-bit support. It was replaced mostly with UINT32, but also ACPI_SIZE where a type that changes 32/64 bit on
Eliminated the ACPI_NATIVE_UINT type across all ACPICA code. This type is no longer needed, especially with the removal of 16-bit support. It was replaced mostly with UINT32, but also ACPI_SIZE where a type that changes 32/64 bit on 32/64-bit platforms is required.

Added the C const qualifier for appropriate string constants -- mostly MODULE_NAME and printf format strings. Jan Beulich.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has a**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.0K Code, 17.4K Data,  97.4K Total
    Debug Version:     159.4K Code, 64.4K Data, 223.8K Total
  Current Release:
    Non-Debug Version:  79.5K Code, 16.2K Data,  95.7K Total
    Debug Version:     153.3K Code, 48.3K Data, 201.6K Total
```

2) iASL Compiler/Disassembler and Tools:

Implemented ACPI table revision ID validation in the disassembler. Zero is always invalid. For DSDTs, the ID controls the interpreter integer width. 1
Implemented ACPI table revision ID validation in the disassembler. Zero is always invalid. For DSDTs, the ID controls the interpreter integer width. 1 means 32-bit and this is unusual. 2 or greater is 64-bit.

---

21 March 2008. Summary of changes for version 20080321:

1) ACPI CA Core Subsystem:

Implemented an additional change to the GPE support in order to suppress **spurious or stray GPEs. The AcpiEvDisableGpe function will now permanently disable incoming GPEs that are neither enabled nor disabled -- meaning that the GPE is unknown to the system. This should prevent future interrupt floods**
spurious or stray GPEs. The AcpiEvDisableGpe function will now permanently disable incoming GPEs that are neither enabled nor disabled -- meaning that the GPE is unknown to the system. This should prevent future interrupt floods from that GPE. BZ 6217 (Zhang Rui)

Fixed a problem where NULL package elements were not returned to the AcpiEvaluateObject interface correctly. The element was simply ignored **instead of returning a NULL ACPI_OBJECT package element, potentially causing a buffer overflow and/or confusing the caller who expected a fixed number of**
instead of returning a NULL ACPI_OBJECT package element, potentially causing a buffer overflow and/or confusing the caller who expected a fixed number of elements. BZ 10132 (Lin Ming, Bob Moore)

Fixed a problem with the CreateField, CreateXXXField (Bit, Byte, Word, Dword, Qword), Field, BankField, and IndexField operators when invoked from inside an executing control method. In this case, these operators created namespace
Fixed a problem with the CreateField, CreateXXXField (Bit, Byte, Word, Dword, Qword), Field, BankField, and IndexField operators when invoked from inside an executing control method. In this case, these operators created namespace nodes that were incorrectly left marked as permanent nodes instead of temporary nodes. This could cause a problem if there is race condition **between an exiting control method and a running namespace walk. (Reported by**
between an exiting control method and a running namespace walk. (Reported by Linn Crosetto)

Fixed a problem where the CreateField and CreateXXXField operators would **incorrectly allow duplicate names (the name of the field) with no exception** *incorrectly allow duplicate names (the name of the field) with no exception* generated.

Implemented several changes for Notify handling. Added support for new Notify *Implemented several changes for Notify handling. Added support for new Notify* values (ACPI 2.0+) and improved the Notify debug output. Notify on **PowerResource objects is no longer allowed, as per the ACPI specification.** *PowerResource objects is no longer allowed, as per the ACPI specification.* (Bob Moore, Zhang Rui)

All Reference Objects returned via the AcpiEvaluateObject interface are now marked as type "REFERENCE" instead of "ANY". The type ANY is now reserved for NULL objects - either NULL package elements or unresolved named references. *All Reference Objects returned via the AcpiEvaluateObject interface are now marked as type "REFERENCE" instead of "ANY". The type ANY is now reserved for NULL objects - either NULL package elements or unresolved named references.*

Fixed a problem where an extraneous debug message was produced for package *Fixed a problem where an extraneous debug message was produced for package* objects (when debugging enabled). The message "Package List length larger **than NumElements count" is now produced in the correct case, and is now an** *than NumElements count" is now produced in the correct case, and is now an* error message rather than a debug message. Added a debug message for the **opposite case, where NumElements is larger than the Package List (the package** *opposite case, where NumElements is larger than the Package List (the package* will be padded out with NULL elements as per the ACPI spec.)

Implemented several improvements for the output of the ASL "Debug" object to *Implemented several improvements for the output of the ASL "Debug" object to* clarify and keep all data for a given object on one output line.

Fixed two size calculation issues with the variable-length Start Dependent *Fixed two size calculation issues with the variable-length Start Dependent* resource descriptor.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has** *debug version of the code includes the debug output trace mechanism and has* a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.7K Code, 17.3K Data,  97.0K Total
    Debug Version:     158.9K Code, 64.0K Data, 222.9K Total
  Current Release:
    Non-Debug Version:  80.0K Code, 17.4K Data,  97.4K Total
    Debug Version:     159.4K Code, 64.4K Data, 223.8K Total
```

2) iASL Compiler/Disassembler and Tools:

Fixed a problem with the use of the Switch operator where execution of

the *Fixed a problem with the use of the Switch operator where execution of the* containing method by multiple concurrent threads could cause an AE_ALREADY_EXISTS exception. This is caused by the fact that there is no actual Switch opcode, it must be simulated with local named temporary **variables and if/else pairs. The solution chosen was to mark any method that uses Switch as Serialized, thus preventing multiple thread entries. BZ 469.** *variables and if/else pairs. The solution chosen was to mark any method that uses Switch as Serialized, thus preventing multiple thread entries. BZ 469.*

---

13 February 2008. Summary of changes for version 20080213:

1) ACPI CA Core Subsystem:

Implemented another MS compatibility design change for GPE/Notify handling. GPEs are now cleared/enabled asynchronously to allow all pending notifies to *Implemented another MS compatibility design change for GPE/Notify handling. GPEs are now cleared/enabled asynchronously to allow all pending notifies to* complete first. It is expected that the OSL will queue the enable request **behind all pending notify requests (may require changes to the local host OSL** *behind all pending notify requests (may require changes to the local host OSL* in AcpiOsExecute). Alexey Starikovskiy.

Fixed a problem where buffer and package objects passed as arguments to a **control method via the external AcpiEvaluateObject interface could cause an** *control method via the external AcpiEvaluateObject interface could cause an* AE_AML_INTERNAL exception depending on the order and type of operators executed by the target control method.

Fixed a problem where resource descriptor size optimization could cause a **problem when a _CRS resource template is passed to a _SRS method. The _SRS** *problem when a _CRS resource template is passed to a _SRS method. The _SRS* resource template must use the same descriptors (with the same size) as **returned from _CRS. This change affects the following resource descriptors: IRQ / IRQNoFlags and StartDependendentFn / StartDependentFnNoPri. (BZ 9487)** *returned from _CRS. This change affects the following resource descriptors: IRQ / IRQNoFlags and StartDependendentFn / StartDependentFnNoPri. (BZ 9487)*

Fixed a problem where a CopyObject to RegionField, BankField, and IndexField objects did not perform an implicit conversion as it should. These types must retain their initial type permanently as per the ACPI specification. However, *Fixed a problem where a CopyObject to RegionField, BankField, and IndexField objects did not perform an implicit conversion as it should. These types must retain their initial type permanently as per the ACPI specification. However,* a CopyObject to all other object types should not perform an implicit conversion, as per the ACPI specification. (Lin Ming, Bob Moore) BZ 388

Fixed a problem with the AcpiGetDevices interface where the mechanism to match device CIDs did not examine the entire list of available CIDs, but instead aborted on the first non-matching CID. Andrew Patterson.

Fixed a regression introduced in version 20071114. The ACPI_HIDWORD macro was

Fixed a regression introduced in version 20071114. The ACPI_HIDWORD macro was
inadvertently changed to return a 16-bit value instead of a 32-bit value,
truncating the upper dword of a 64-bit value. This macro is only used to
display debug output, so no incorrect calculations were made. Also,
reimplemented the macro so that a 64-bit shift is not performed by
inefficient compilers.

Added missing va_end statements that should correspond with each va_start
statement.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.5K Code, 17.2K Data,  96.7K Total
    Debug Version:     159.0K Code, 63.8K Data, 222.8K Total
  Current Release:
    Non-Debug Version:  79.7K Code, 17.3K Data,  97.0K Total
    Debug Version:     158.9K Code, 64.0K Data, 222.9K Total
```

2) iASL Compiler/Disassembler and Tools:

Implemented full disassembler support for the following new ACPI tables:
BERT, EINJ, and ERST. Implemented partial disassembler support for the
complicated HEST table. These tables support the Windows Hardware Error
Architecture (WHEA).

---
23 January 2008. Summary of changes for version 20080123:

1) ACPI CA Core Subsystem:

Added the 2008 copyright to all module headers and signons. This affects
virtually every file in the ACPICA core subsystem, the iASL compiler, and
the tools/utilities.

Fixed a problem with the SizeOf operator when used with Package and
Buffer
objects. These objects have deferred execution for some arguments, and
the
execution is now completed before the SizeOf is executed. This problem
caused
unexpected AE_PACKAGE_LIMIT errors on some systems (Lin Ming, Bob Moore)
BZ
Fixed a problem with the SizeOf operator when used with Package and Buffer
objects. These objects have deferred execution for some arguments, and the
execution is now completed before the SizeOf is executed. This problem caused
unexpected AE_PACKAGE_LIMIT errors on some systems (Lin Ming, Bob Moore) BZ
9558

Implemented an enhancement to the interpreter "slack mode". In the
absence
of
an explicit return or an implicitly returned object from the last
executed
opcode, a control method will now implicitly return an integer of value 0
for
Implemented an enhancement to the interpreter "slack mode". In the absence of
an explicit return or an implicitly returned object from the last executed
opcode, a control method will now implicitly return an integer of value 0 for
Microsoft compatibility. (Lin Ming) BZ 392

Fixed a problem with the Load operator where an exception was not
returned
in
Fixed a problem with the Load operator where an exception was not returned in
the case where the table is already loaded. (Lin Ming) BZ 463

Implemented support for the use of DDBHandles as an Indexed Reference, as
per
Implemented support for the use of DDBHandles as an Indexed Reference, as per
the ACPI spec. (Lin Ming) BZ 486

Implemented support for UserTerm (Method invocation) for the Unload
operator
Implemented support for UserTerm (Method invocation) for the Unload operator
as per the ACPI spec. (Lin Ming) BZ 580

Fixed a problem with the LoadTable operator where the OemId and
OemTableId
input strings could cause unexpected failures if they were shorter than
the
Fixed a problem with the LoadTable operator where the OemId and OemTableId
input strings could cause unexpected failures if they were shorter than the
maximum lengths allowed. (Lin Ming, Bob Moore) BZ 576

Implemented support for UserTerm (Method invocation) for the Unload
operator
Implemented support for UserTerm (Method invocation) for the Unload operator
as per the ACPI spec. (Lin Ming) BZ 580

Implemented header file support for new ACPI tables - BERT, ERST, EINJ,
HEST,
Implemented header file support for new ACPI tables - BERT, ERST, EINJ, HEST,
IBFT, UEFI, WDAT. Disassembler support is forthcoming.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.3K Code, 17.2K Data,  96.5K Total
    Debug Version:     158.6K Code, 63.8K Data, 222.4K Total
  Current Release:
    Non-Debug Version:  79.5K Code, 17.2K Data,  96.7K Total
    Debug Version:     159.0K Code, 63.8K Data, 222.8K Total
```

2) iASL Compiler/Disassembler and Tools:

Implemented support in the disassembler for checksum validation on
incoming
binary DSDTs and SSDTs. If incorrect, a message is displayed within the
table
Implemented support in the disassembler for checksum validation on incoming
binary DSDTs and SSDTs. If incorrect, a message is displayed within the table
header dump at the start of the disassembly.

Implemented additional debugging information in the namespace listing
file
created during compilation. In addition to the namespace hierarchy, the
full
Implemented additional debugging information in the namespace listing file
created during compilation. In addition to the namespace hierarchy, the full
pathname to each namespace object is displayed.

Fixed a problem with the disassembler where invalid ACPI tables could cause
Fixed a problem with the disassembler where invalid ACPI tables could cause faults or infinite loops.

Fixed an unexpected parse error when using the optional "parameter types" list in a control method declaration. (Lin Ming) BZ 397

Fixed a problem where two External declarations with the same name did not
Fixed a problem where two External declarations with the same name did not cause an error (Lin Ming) BZ 509

Implemented support for full TermArgs (adding Argx, Localx and method **invocation) for the ParameterData parameter to the LoadTable operator. (Lin**
invocation) for the ParameterData parameter to the LoadTable operator. (Lin Ming) BZ 583,587

---
19 December 2007. Summary of changes for version 20071219:

1) ACPI CA Core Subsystem:

Implemented full support for deferred execution for the TermArg string arguments for DataTableRegion. This enables forward references and full **operand resolution for the three string arguments. Similar to OperationRegion**
operand resolution for the three string arguments. Similar to OperationRegion deferred argument execution.) Lin Ming. BZ 430

Implemented full argument resolution support for the BankValue argument to BankField. Previously, only constants were supported, now any TermArg may be
Implemented full argument resolution support for the BankValue argument to BankField. Previously, only constants were supported, now any TermArg may be used. Lin Ming BZ 387, 393

Fixed a problem with AcpiGetDevices where the search of a branch of the device tree could be terminated prematurely. In accordance with the ACPI **specification, the search down the current branch is terminated if a device is both not present and not functional (instead of just not present.) Yakui**
specification, the search down the current branch is terminated if a device is both not present and not functional (instead of just not present.) Yakui Zhao.

Fixed a problem where "unknown" GPEs could be allowed to fire repeatedly if the underlying AML code changed the GPE enable registers. Now, any unknown incoming GPE (no _Lxx/_Exx method and not the EC GPE) is immediately disabled
Fixed a problem where "unknown" GPEs could be allowed to fire repeatedly if the underlying AML code changed the GPE enable registers. Now, any unknown incoming GPE (no _Lxx/_Exx method and not the EC GPE) is immediately disabled instead of simply ignored. Rui Zhang.

Fixed a problem with Index Fields where the Index register was incorrectly
Fixed a problem with Index Fields where the Index register was incorrectly limited to a maximum of 32 bits. Now any size may be used.

Fixed a couple memory leaks associated with "implicit return" objects when
Fixed a couple memory leaks associated with "implicit return" objects when the AML Interpreter slack mode is enabled. Lin Ming BZ 349

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has**
debug version of the code includes the debug output trace mechanism and has a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.0K Code, 17.2K Data,  96.2K Total
    Debug Version:     157.9K Code, 63.6K Data, 221.5K Total
  Current Release:
    Non-Debug Version:  79.3K Code, 17.2K Data,  96.5K Total
    Debug Version:     158.6K Code, 63.8K Data, 222.4K Total
```

---
14 November 2007. Summary of changes for version 20071114:

1) ACPI CA Core Subsystem:

Implemented event counters for each of the Fixed Events, the ACPI SCI (interrupt) itself, and control methods executed. Named **AcpiFixedEventCount[], AcpiSciCount, and AcpiMethodCount respectively. These**
AcpiFixedEventCount[], AcpiSciCount, and AcpiMethodCount respectively. These should be useful for debugging and statistics.

Implemented a new external interface, AcpiGetStatistics, to retrieve the contents of the various event counters. Returns the current values for AcpiSciCount, AcpiGpeCount, the AcpiFixedEventCount array, and **AcpiMethodCount. The interface can be expanded in the future if new counters are added. Device drivers should use this interface rather than access the**
AcpiMethodCount. The interface can be expanded in the future if new counters are added. Device drivers should use this interface rather than access the counters directly.

Fixed a problem with the FromBCD and ToBCD operators. With some compilers, the ShortDivide function worked incorrectly, causing problems with the BCD
Fixed a problem with the FromBCD and ToBCD operators. With some compilers, the ShortDivide function worked incorrectly, causing problems with the BCD functions with large input values. A truncation from 64-bit to 32-bit inadvertently occurred. Internal BZ 435. Lin Ming

Fixed a problem with Index references passed as method arguments. References passed as arguments to control methods were dereferenced immediately (before control was passed to the called method). The references are now correctly
Fixed a problem with Index references passed as method arguments. References passed as arguments to control methods were dereferenced immediately (before control was passed to the called method). The references are now correctly passed directly to the called method. BZ 5389. Lin Ming

Fixed a problem with CopyObject used in conjunction with the Index operator. The reference was incorrectly dereferenced before the copy. The reference is
Fixed a problem with CopyObject used in conjunction with the Index operator.

The reference was incorrectly dereferenced before the copy. The reference is
now correctly copied. BZ 5391. Lin Ming

Fixed a problem with Control Method references within Package objects.
These
Fixed a problem with Control Method references within Package objects. These
references are now correctly generated. This completes the package
construction overhaul that began in version 20071019.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.8K Code, 17.2K Data,  96.0K Total
    Debug Version:     157.2K Code, 63.4K Data, 220.6K Total
  Current Release:
    Non-Debug Version:  79.0K Code, 17.2K Data,  96.2K Total
    Debug Version:     157.9K Code, 63.6K Data, 221.5K Total
```

2) iASL Compiler/Disassembler and Tools:

The AcpiExec utility now installs handlers for all of the predefined
Operation Region types. New types supported are: PCI_Config, CMOS, and
PCIBARTarget.

Fixed a problem with the 64-bit version of AcpiExec where the extended
(64-
Fixed a problem with the 64-bit version of AcpiExec where the extended (64-
bit) address fields for the DSDT and FACS within the FADT were not being
used, causing truncation of the upper 32-bits of these addresses. Lin
Ming
used, causing truncation of the upper 32-bits of these addresses. Lin Ming
and Bob Moore

---
19 October 2007. Summary of changes for version 20071019:

1) ACPI CA Core Subsystem:

Fixed a problem with the Alias operator when the target of the alias is a
named ASL operator that opens a new scope -- Scope, Device,
PowerResource,
named ASL operator that opens a new scope -- Scope, Device, PowerResource,
Processor, and ThermalZone. In these cases, any children of the original
operator could not be accessed via the alias, potentially causing
unexpected
operator could not be accessed via the alias, potentially causing unexpected
AE_NOT_FOUND exceptions. (BZ 9067)

Fixed a problem with the Package operator where all named references were
created as object references and left otherwise unresolved. According to
the
ACPI specification, a Package can only contain Data Objects or references
to
created as object references and left otherwise unresolved. According to the
ACPI specification, a Package can only contain Data Objects or references to
control methods. The implication is that named references to Data Objects
(Integer, Buffer, String, Package, BufferField, Field) should be resolved
immediately upon package creation. This is the approach taken with this
change. References to all other named objects (Methods, Devices, Scopes,
etc.) are all now properly created as reference objects. (BZ 5328)

Reverted a change to Notify handling that was introduced in version
20070508. This version changed the Notify handling from asynchronous to
fully synchronous (Device driver Notify handling with respect to the
Notify
fully synchronous (Device driver Notify handling with respect to the Notify
ASL operator). It was found that this change caused more problems than it
solved and was removed by most users.

Fixed a problem with the Increment and Decrement operators where the type
of
Fixed a problem with the Increment and Decrement operators where the type of
the target object could be unexpectedly and incorrectly changed. (BZ 353)
Lin Ming.

Fixed a problem with the Load and LoadTable operators where the table
location within the namespace was ignored. Instead, the table was always
loaded into the root or current scope. Lin Ming.

Fixed a problem with the Load operator when loading a table from a buffer
object. The input buffer was prematurely zeroed and/or deleted. (BZ 577)

Fixed a problem with the Debug object where a store of a DdbHandle
reference
Fixed a problem with the Debug object where a store of a DdbHandle reference
object to the Debug object could cause a fault.

Added a table checksum verification for the Load operator, in the case
where
Added a table checksum verification for the Load operator, in the case where
the load is from a buffer. (BZ 578).

Implemented additional parameter validation for the LoadTable operator.
The
length of the input strings SignatureString, OemIdString, and OemTableId
are
Implemented additional parameter validation for the LoadTable operator. The
length of the input strings SignatureString, OemIdString, and OemTableId are
now checked for maximum lengths. (BZ 582) Lin Ming.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.5K Code, 17.1K Data,  95.6K Total
    Debug Version:     156.7K Code, 63.2K Data, 219.9K Total
  Current Release:
    Non-Debug Version:  78.8K Code, 17.2K Data,  96.0K Total
    Debug Version:     157.2K Code, 63.4K Data, 220.6K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem where if a single file was specified and the file did not
exist, no error message was emitted. (Introduced with wildcard support in
version 20070917.)

---
19 September 2007. Summary of changes for version 20070919:

1) ACPI CA Core Subsystem:

Designed and implemented new external interfaces to install and remove
**handlers for ACPI table-related events. Current events that are defined
are**
handlers for ACPI table-related events. Current events that are defined are
LOAD and UNLOAD. These interfaces allow the host to track ACPI tables as
they are dynamically loaded and unloaded. See AcpiInstallTableHandler and
AcpiRemoveTableHandler. (Lin Ming and Bob Moore)

Fixed a problem where the use of the AcpiGbl_AllMethodsSerialized flag
(acpi_serialized option on Linux) could cause some systems to hang during
initialization. (Bob Moore) BZ 8171

Fixed a problem where objects of certain types (Device, ThermalZone,
Processor, PowerResource) can be not found if they are declared and
referenced from within the same control method (Lin Ming) BZ 341

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
**debug version of the code includes the debug output trace mechanism and
has**
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.3K Code, 17.0K Data,  95.3K Total
    Debug Version:     156.3K Code, 63.1K Data, 219.4K Total
  Current Release:
    Non-Debug Version:  78.5K Code, 17.1K Data,  95.6K Total
    Debug Version:     156.7K Code, 63.2K Data, 219.9K Total
```

2) iASL Compiler/Disassembler:

**Implemented support to allow multiple files to be compiled/disassembled
in
a
single invocation. This includes command line wildcard support for both
the**
Implemented support to allow multiple files to be compiled/disassembled in a
single invocation. This includes command line wildcard support for both the
Windows and Unix versions of the compiler. This feature simplifies the
**disassembly and compilation of multiple ACPI tables in a single
directory.**
disassembly and compilation of multiple ACPI tables in a single directory.

---

08 May 2007. Summary of changes for version 20070508:

1) ACPI CA Core Subsystem:

**Implemented a Microsoft compatibility design change for the handling of
the**
Implemented a Microsoft compatibility design change for the handling of the
Notify AML operator. Previously, notify handlers were dispatched and
executed completely asynchronously in a deferred thread. The new design
**still executes the notify handlers in a different thread, but the
original
thread that executed the Notify() now waits at a synchronization point
for
the notify handler to complete. Some machines depend on a synchronous
Notify**
still executes the notify handlers in a different thread, but the original
thread that executed the Notify() now waits at a synchronization point for
the notify handler to complete. Some machines depend on a synchronous Notify
operator in order to operate correctly.

Implemented support to allow Package objects to be passed as method
arguments to the external AcpiEvaluateObject interface. Previously, this
would return the AE_NOT_IMPLEMENTED exception. This feature had not been
implemented since there were no reserved control methods that required it
until recently.

**Fixed a problem with the internal FADT conversion where ACPI 1.0 FADTs
that**
Fixed a problem with the internal FADT conversion where ACPI 1.0 FADTs that
contained invalid non-zero values in reserved fields could cause later
**failures because these fields have meaning in later revisions of the
FADT.
For incoming ACPI 1.0 FADTs, these fields are now always zeroed. (The
fields**
failures because these fields have meaning in later revisions of the FADT.
For incoming ACPI 1.0 FADTs, these fields are now always zeroed. (The fields
are: Preferred_PM_Profile, PSTATE_CNT, CST_CNT, and IAPC_BOOT_FLAGS.)

**Fixed a problem where the Global Lock handle was not properly updated if
a
thread that acquired the Global Lock via executing AML code then
attempted
to acquire the lock via the AcpiAcquireGlobalLock interface. Reported by
Joe**
Fixed a problem where the Global Lock handle was not properly updated if a
thread that acquired the Global Lock via executing AML code then attempted
to acquire the lock via the AcpiAcquireGlobalLock interface. Reported by Joe
Liu.

Fixed a problem in AcpiEvDeleteGpeXrupt where the global interrupt list
could be corrupted if the interrupt being removed was at the head of the
list. Reported by Linn Crosetto.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
**debug version of the code includes the debug output trace mechanism and
has**
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.9K Code, 63.1K Data, 219.0K Total
  Current Release:
    Non-Debug Version:  78.3K Code, 17.0K Data,  95.3K Total
    Debug Version:     156.3K Code, 63.1K Data, 219.4K Total
```

---

20 March 2007. Summary of changes for version 20070320:

1) ACPI CA Core Subsystem:

Implemented a change to the order of interpretation and evaluation of AML
operand objects within the AML interpreter. The interpreter now evaluates
operands in the order that they appear in the AML stream (and the
**corresponding ASL code), instead of in the reverse order (after the
entire
operand list has been parsed). The previous behavior caused several
subtle**
corresponding ASL code), instead of in the reverse order (after the entire
operand list has been parsed). The previous behavior caused several subtle
incompatibilities with the Microsoft AML interpreter as well as being
somewhat non-intuitive. BZ 7871, local BZ 263. Valery Podrezov.

**Implemented a change to the ACPI Global Lock support. All interfaces to
the**

Implemented a change to the ACPI Global Lock support. All interfaces to the global lock now allow the same thread to acquire the lock multiple times. **This affects the AcpiAcquireGlobalLock external interface to the global lock** *This affects the AcpiAcquireGlobalLock external interface to the global lock* as well as the internal use of the global lock to support AML fields -- a **control method that is holding the global lock can now simultaneously access AML fields that require global lock protection. Previously, in both cases, this would have resulted in an AE_ALREADY_ACQUIRED exception. The change to** *control method that is holding the global lock can now simultaneously access AML fields that require global lock protection. Previously, in both cases, this would have resulted in an AE_ALREADY_ACQUIRED exception. The change to* AcpiAcquireGlobalLock is of special interest to drivers for the Embedded **Controller. There is no change to the behavior of the AML Acquire operator,** *Controller. There is no change to the behavior of the AML Acquire operator,* as this can already be used to acquire a mutex multiple times by the same thread. BZ 8066. With assistance from Alexey Starikovskiy.

Fixed a problem where invalid objects could be referenced in the AML **Interpreter after error conditions. During operand evaluation, ensure that** *Interpreter after error conditions. During operand evaluation, ensure that* the internal "Return Object" field is cleared on error and only valid **pointers are stored there. Caused occasional access to deleted objects that** *pointers are stored there. Caused occasional access to deleted objects that* resulted in "large reference count" warning messages. Valery Podrezov.

Fixed a problem where an AE_STACK_OVERFLOW internal exception could occur on *Fixed a problem where an AE_STACK_OVERFLOW internal exception could occur on* deeply nested control method invocations. BZ 7873, local BZ 487. Valery Podrezov.

Fixed an internal problem with the handling of result objects on the interpreter result stack. BZ 7872. Valery Podrezov.

Removed obsolete code that handled the case where AML_NAME_OP is the target *Removed obsolete code that handled the case where AML_NAME_OP is the target* of a reference (Reference.Opcode). This code was no longer necessary. BZ 7874. Valery Podrezov.

Removed obsolete ACPI_NO_INTEGER64_SUPPORT from two header files. This was a *Removed obsolete ACPI_NO_INTEGER64_SUPPORT from two header files. This was a* remnant from the previously discontinued 16-bit support.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has** *debug version of the code includes the debug output trace mechanism and has* a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.8K Code, 63.3K Data, 219.1K Total
  Current Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.9K Code, 63.1K Data, 219.0K Total
```


---

26 January 2007. Summary of changes for version 20070126:

1) ACPI CA Core Subsystem:

Added the 2007 copyright to all module headers and signons. This affects virtually every file in the ACPICA core subsystem, the iASL compiler, and the utilities.

Implemented a fix for an incorrect parameter passed to AcpiTbDeleteTable **during a table load. A bad pointer was passed in the case where the DSDT is** *during a table load. A bad pointer was passed in the case where the DSDT is* overridden, causing a fault in this case.

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has** *debug version of the code includes the debug output trace mechanism and has* a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.8K Code, 63.3K Data, 219.1K Total
  Current Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.8K Code, 63.3K Data, 219.1K Total
```

---

15 December 2006. Summary of changes for version 20061215:

1) ACPI CA Core Subsystem:

Support for 16-bit ACPICA has been completely removed since it is no longer *Support for 16-bit ACPICA has been completely removed since it is no longer* necessary and it clutters the code. All 16-bit macros, types, and **conditional compiles have been removed, cleaning up and simplifying the code** *conditional compiles have been removed, cleaning up and simplifying the code* across the entire subsystem. DOS support is no longer needed since the bootable Linux firmware kit is now available.

The handler for the Global Lock is now removed during AcpiTerminate to enable a clean subsystem restart, via the implementation of the AcpiEvRemoveGlobalLockHandler function. (With assistance from Joel Bretz, HP)

Implemented enhancements to the multithreading support within the debugger to enable improved multithreading debugging and evaluation of the subsystem. *Implemented enhancements to the multithreading support within the debugger to enable improved multithreading debugging and evaluation of the subsystem.* (Valery Podrezov)

Debugger: Enhanced the Statistics/Memory command to emit the total (maximum) memory used during the execution, as well as the maximum memory consumed by *Debugger: Enhanced the Statistics/Memory command to emit the total (maximum) memory used during the execution, as well as the maximum memory consumed by* each of the various object types. (Valery Podrezov)

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  77.9K Code, 17.0K Data,  94.9K Total
    Debug Version:     155.2K Code, 63.1K Data, 218.3K Total
  Current Release:
    Non-Debug Version:  78.0K Code, 17.1K Data,  95.1K Total
    Debug Version:     155.8K Code, 63.3K Data, 219.1K Total
```

2) iASL Compiler/Disassembler and Tools:

AcpiExec: Implemented a new option (-m) to display full memory use
statistics upon subsystem/program termination. (Valery Podrezov)

---
09 November 2006. Summary of changes for version 20061109:

1) ACPI CA Core Subsystem:

Optimized the Load ASL operator in the case where the source operand is
an
Optimized the Load ASL operator in the case where the source operand is an
operation region. Simply map the operation region memory, instead of
performing a bytewise read. (Region must be of type SystemMemory, see
below.)

Fixed the Load ASL operator for the case where the source operand is a
region field. A buffer object is also allowed as the source operand. BZ
480
region field. A buffer object is also allowed as the source operand. BZ 480

Fixed a problem where the Load ASL operator allowed the source operand to
be
Fixed a problem where the Load ASL operator allowed the source operand to be
an operation region of any type. It is now restricted to regions of type
SystemMemory, as per the ACPI specification. BZ 481

Additional cleanup and optimizations for the new Table Manager code.

AcpiEnable will now fail if all of the required ACPI tables are not
loaded
AcpiEnable will now fail if all of the required ACPI tables are not loaded
(FADT, FACS, DSDT). BZ 477

Added #pragma pack(8/4) to acobject.h to ensure that the structures in
this
Added #pragma pack(8/4) to acobject.h to ensure that the structures in this
header are always compiled as aligned. The ACPI_OPERAND_OBJECT has been
manually optimized to be aligned and will not work if it is byte-packed.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  78.1K Code, 17.1K Data,  95.2K Total
    Debug Version:     155.4K Code, 63.1K Data, 218.5K Total
```



```
  Current Release:
    Non-Debug Version:  77.9K Code, 17.0K Data,  94.9K Total
    Debug Version:     155.2K Code, 63.1K Data, 218.3K Total
```

2) iASL Compiler/Disassembler and Tools:

Fixed a problem where the presence of the _OSI predefined control method
within complex expressions could cause an internal compiler error.

AcpiExec: Implemented full region support for multiple address spaces.
SpaceId is now part of the REGION object. BZ 429

---
11 October 2006. Summary of changes for version 20061011:

1) ACPI CA Core Subsystem:

Completed an AML interpreter performance enhancement for control method
execution. Previously a 2-pass parse/execution, control methods are now
completely parsed and executed in a single pass. This improves overall
interpreter performance by ~25%, reduces code size, and reduces CPU stack
use. (Valery Podrezov + interpreter changes in version 20051202 that
eliminated namespace loading during the pass one parse.)

Implemented _CID support for PCI Root Bridge detection. If the _HID does
not
match the predefined PCI Root Bridge IDs, the _CID list (if present) is
now
Implemented _CID support for PCI Root Bridge detection. If the _HID does not
match the predefined PCI Root Bridge IDs, the _CID list (if present) is now
obtained and also checked for an ID match.

Implemented additional support for the PCI _ADR execution: upsearch until
a
Implemented additional support for the PCI _ADR execution: upsearch until a
device scope is found before executing _ADR. This allows PCI_Config
operation regions to be declared locally within control methods
underneath
operation regions to be declared locally within control methods underneath
PCI device objects.

Fixed a problem with a possible race condition between threads executing
AcpiWalkNamespace and the AML interpreter. This condition was removed by
modifying AcpiWalkNamespace to (by default) ignore all temporary
namespace
modifying AcpiWalkNamespace to (by default) ignore all temporary namespace
entries created during any concurrent control method execution. An
additional namespace race condition is known to exist between
AcpiWalkNamespace and the Load/Unload ASL operators and is still under
investigation.

Restructured the AML ParseLoop function, breaking it into several
subfunctions in order to reduce CPU stack use and improve
maintainability.
subfunctions in order to reduce CPU stack use and improve maintainability.
(Mikhail Kouzmich)

AcpiGetHandle: Fix for parameter validation to detect invalid
combinations
AcpiGetHandle: Fix for parameter validation to detect invalid combinations
of prefix handle and pathname. BZ 478

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and

has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  77.9K Code, 17.1K Data,  95.0K Total
    Debug Version:     154.6K Code, 63.0K Data, 217.6K Total
  Current Release:
    Non-Debug Version:  78.1K Code, 17.1K Data,  95.2K Total
    Debug Version:     155.4K Code, 63.1K Data, 218.5K Total
```

2) iASL Compiler/Disassembler and Tools:

Ported the -g option (get local ACPI tables) to the new ACPICA Table Manager
Ported the -g option (get local ACPI tables) to the new ACPICA Table Manager
to restore original behavior.

---
27 September 2006. Summary of changes for version 20060927:

1) ACPI CA Core Subsystem:

Removed the "Flags" parameter from AcpiGetRegister and AcpiSetRegister.
These functions now use a spinlock for mutual exclusion and the interrupt
level indication flag is not needed.

Fixed a problem with the Global Lock where the lock could appear to be
obtained before it is actually obtained. The global lock semaphore was
inadvertently created with one unit instead of zero units. (BZ 464) Fiodor
inadvertently created with one unit instead of zero units. (BZ 464) Fiodor
Suietov.

Fixed a possible memory leak and fault in AcpiExResolveObjectToValue during
Fixed a possible memory leak and fault in AcpiExResolveObjectToValue during
a read from a buffer or region field. (BZ 458) Fiodor Suietov.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  77.9K Code, 17.1K Data,  95.0K Total
    Debug Version:     154.7K Code, 63.0K Data, 217.7K Total
  Current Release:
    Non-Debug Version:  77.9K Code, 17.1K Data,  95.0K Total
    Debug Version:     154.6K Code, 63.0K Data, 217.6K Total
```

2) iASL Compiler/Disassembler and Tools:

Fixed a compilation problem with the pre-defined Resource Descriptor field names where an "object does not exist" error could be incorrectly generated
Fixed a compilation problem with the pre-defined Resource Descriptor field names where an "object does not exist" error could be incorrectly generated
if the parent ResourceTemplate pathname places the template within a
different namespace scope than the current scope. (BZ 7212)

Fixed a problem where the compiler could hang after syntax errors

detected
Fixed a problem where the compiler could hang after syntax errors detected
in an ElseIf construct. (BZ 453)

Fixed a problem with the AmlFilename parameter to the DefinitionBlock()
operator. An incorrect output filename was produced when this parameter was
operator. An incorrect output filename was produced when this parameter was
a null string (""). Now, the original input filename is used as the AML
output filename, with an ".aml" extension.

Implemented a generic batch command mode for the AcpiExec utility (execute
Implemented a generic batch command mode for the AcpiExec utility (execute
any AML debugger command) (Valery Podrezov).

---
12 September 2006. Summary of changes for version 20060912:

1) ACPI CA Core Subsystem:

Enhanced the implementation of the "serialized mode" of the interpreter
(enabled via the AcpiGbl_AllMethodsSerialized flag.) When this mode is
specified, instead of creating a serialization semaphore per control method,
specified, instead of creating a serialization semaphore per control method,
the interpreter lock is simply no longer released before a blocking
operation during control method execution. This effectively makes the AML
Interpreter single-threaded. The overhead of a semaphore per-method is
eliminated.

Fixed a regression where an error was no longer emitted if a control method
Fixed a regression where an error was no longer emitted if a control method
attempts to create 2 objects of the same name. This once again returns
AE_ALREADY_EXISTS. When this exception occurs, it invokes the mechanism that
AE_ALREADY_EXISTS. When this exception occurs, it invokes the mechanism that
will dynamically serialize the control method to possible prevent future
errors. (BZ 440)

Integrated a fix for a problem with PCI Express HID detection in the PCI
Config Space setup procedure. (BZ 7145)

Moved all FADT-related functions to a new file, tbfadt.c. Eliminated the
AcpiHwInitialize function - the FADT registers are now validated when the
table is loaded.

Added two new warnings during FADT verification - 1) if the FADT is larger than the largest known FADT version, and 2) if there is a mismatch between a 32-bit block address and the 64-bit X counterpart (when both are non-zero.)
Added two new warnings during FADT verification - 1) if the FADT is larger than the largest known FADT version, and 2) if there is a mismatch between a 32-bit block address and the 64-bit X counterpart (when both are non-zero.)

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  77.9K Code, 16.7K Data,  94.6K Total
    Debug Version:     154.9K Code, 62.6K Data, 217.5K Total
  Current Release:
    Non-Debug Version:  77.9K Code, 17.1K Data,  95.0K Total
    Debug Version:     154.7K Code, 63.0K Data, 217.7K Total


2) iASL Compiler/Disassembler and Tools:

Fixed a problem with the implementation of the Switch() operator where
the
temporary variable was declared too close to the actual Switch, instead
of
Fixed a problem with the implementation of the Switch() operator where the
temporary variable was declared too close to the actual Switch, instead of
at method level. This could cause a problem if the Switch() operator is
within a while loop, causing an error on the second iteration. (BZ 460)

Disassembler - fix for error emitted for unknown type for target of scope
operator. Now, ignore it and continue.

Disassembly of an FADT now verifies the input FADT and reports any errors
found. Fix for proper disassembly of full-sized (ACPI 2.0) FADTs.

Disassembly of raw data buffers with byte initialization data now
prefixes
Disassembly of raw data buffers with byte initialization data now prefixes
each output line with the current buffer offset.

Disassembly of ASF! table now includes all variable-length data fields at
the end of some of the subtables.

The disassembler now emits a comment if a buffer appears to be a
ResourceTemplate, but cannot be disassembled as such because the EndTag
does
ResourceTemplate, but cannot be disassembled as such because the EndTag does
not appear at the very end of the buffer.

AcpiExec - Added the "-t" command line option to enable the serialized
mode
AcpiExec - Added the "-t" command line option to enable the serialized mode
of the AML interpreter.

---------------------------------------
31 August 2006. Summary of changes for version 20060831:

1) ACPI CA Core Subsystem:

Miscellaneous fixes for the Table Manager:
- Correctly initialize internal common FADT for all 64-bit "X" fields
- Fixed a couple table mapping issues during table load
- Fixed a couple alignment issues for IA64
- Initialize input array to zero in AcpiInitializeTables
- Additional parameter validation for AcpiGetTable, AcpiGetTableHeader,
AcpiGetTableByIndex

Change for GPE support: when a "wake" GPE is received, all wake GPEs are
now
Change for GPE support: when a "wake" GPE is received, all wake GPEs are now
immediately disabled to prevent the waking GPE from firing again and to
prevent other wake GPEs from interrupting the wake process.

Added the AcpiGpeCount global that tracks the number of processed GPEs,
to
Added the AcpiGpeCount global that tracks the number of processed GPEs, to
```



```
be used for debugging systems with a large number of ACPI interrupts.

Implemented support for the "DMAR" ACPI table (DMA Redirection Table) in
both the ACPICA headers and the disassembler.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

  Previous Release:
    Non-Debug Version:  77.8K Code, 16.5K Data,  94.3K Total
    Debug Version:     154.6K Code, 62.3K Data, 216.9K Total
  Current Release:
    Non-Debug Version:  77.9K Code, 16.7K Data,  94.6K Total
    Debug Version:     154.9K Code, 62.6K Data, 217.5K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler support for the DMAR ACPI table.

---------------------------------------
23 August 2006. Summary of changes for version 20060823:

1) ACPI CA Core Subsystem:

The Table Manager component has been completely redesigned and
reimplemented. The new design is much simpler, and reduces the overall
code
and data size of the kernel-resident ACPICA by approximately 5%. Also, it
is
reimplemented. The new design is much simpler, and reduces the overall code
and data size of the kernel-resident ACPICA by approximately 5%. Also, it is
now possible to obtain the ACPI tables very early during kernel
initialization, even before dynamic memory management is initialized.
(Alexey Starikovskiy, Fiodor Suietov, Bob Moore)

Obsolete ACPICA interfaces:

- AcpiGetFirmwareTable: Use AcpiGetTable instead (works at early kernel
init
- AcpiGetFirmwareTable: Use AcpiGetTable instead (works at early kernel init
time).
- AcpiLoadTable: Not needed.
- AcpiUnloadTable: Not needed.

New ACPICA interfaces:

- AcpiInitializeTables: Must be called before the table manager can be
used.
- AcpiInitializeTables: Must be called before the table manager can be used.
- AcpiReallocateRootTable: Used to transfer the root table to dynamically
allocated memory after it becomes available.
- AcpiGetTableByIndex: Allows the host to easily enumerate all ACPI
tables
- AcpiGetTableByIndex: Allows the host to easily enumerate all ACPI tables
in the RSDT/XSDT.

Other ACPICA changes:

- AcpiGetTableHeader returns the actual mapped table header, not a copy.
Use
- AcpiGetTableHeader returns the actual mapped table header, not a copy. Use
```

AcpiOsUnmapMemory to free this mapping.
- AcpiGetTable returns the actual mapped table. The mapping is managed
internally and must not be deleted by the caller. Use of this interface
causes no additional dynamic memory allocation.
- AcpiFindRootPointer: Support for physical addressing has been eliminated,
- AcpiFindRootPointer: Support for physical addressing has been eliminated,
it appeared to be unused.
- The interface to AcpiOsMapMemory has changed to be consistent with the
other allocation interfaces.
- The interface to AcpiOsGetRootPointer has changed to eliminate unnecessary
- The interface to AcpiOsGetRootPointer has changed to eliminate unnecessary
parameters.
- ACPI_PHYSICAL_ADDRESS is now 32 bits on 32-bit platforms, 64 bits on 64-
- ACPI_PHYSICAL_ADDRESS is now 32 bits on 32-bit platforms, 64 bits on 64-
bit platforms. Was previously 64 bits on all platforms.
- The interface to the ACPI Global Lock acquire/release macros have changed
- The interface to the ACPI Global Lock acquire/release macros have changed
slightly since ACPICA no longer keeps a local copy of the FACS with a
constructed pointer to the actual global lock.

Porting to the new table manager:

- AcpiInitializeTables: Must be called once, and can be called anytime
during the OS initialization process. It allows the host to specify an area
during the OS initialization process. It allows the host to specify an area
of memory to be used to store the internal version of the RSDT/XSDT (root
table). This allows the host to access ACPI tables before memory management
table). This allows the host to access ACPI tables before memory management
is initialized and running.
- AcpiReallocateRootTable: Can be called after memory management is running
- AcpiReallocateRootTable: Can be called after memory management is running
to copy the root table to a dynamically allocated array, freeing up the
scratch memory specified in the call to AcpiInitializeTables.
- AcpiSubsystemInitialize: This existing interface is independent of the
Table Manager, and does not have to be called before the Table Manager can
Table Manager, and does not have to be called before the Table Manager can
be used, it only must be called before the rest of ACPICA can be used.
- ACPI Tables: Some changes have been made to the names and structure of the
actbl.h and actbl1.h header files and may require changes to existing code.
For example, bitfields have been completely removed because of their lack of
- ACPI Tables: Some changes have been made to the names and structure of the
actbl.h and actbl1.h header files and may require changes to existing code.
For example, bitfields have been completely removed because of their lack of
portability across C compilers.
- Update interfaces to the Global Lock acquire/release macros if local
versions are used. (see acwin.h)

Obsolete files: tbconvrt.c, tbget.c, tbgetall.c, tbrsdt.c

New files: tbfind.c

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has

debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.7K Code, 17.9K Data,  98.6K Total
    Debug Version:     161.0K Code, 65.1K Data, 226.1K Total
  Current Release:
    Non-Debug Version:  77.8K Code, 16.5K Data,  94.3K Total
    Debug Version:     154.6K Code, 62.3K Data, 216.9K Total
```

2) iASL Compiler/Disassembler and Tools:

No changes for this release.

---

21 July 2006. Summary of changes for version 20060721:

1) ACPI CA Core Subsystem:

The full source code for the ASL test suite used to validate the iASL
compiler and the ACPICA core subsystem is being released with the ACPICA
source for the first time. The source is contained in a separate package and
consists of over 1100 files that exercise all ASL/AML operators. The package
should appear on the Intel/ACPI web site shortly. (Valery Podrezov, Fiodor
source for the first time. The source is contained in a separate package and
consists of over 1100 files that exercise all ASL/AML operators. The package
should appear on the Intel/ACPI web site shortly. (Valery Podrezov, Fiodor
Suietov)

Completed a new design and implementation for support of the ACPI Global
Lock. On the OS side, the global lock is now treated as a standard AML
mutex. Previously, multiple OS threads could "acquire" the global lock
simultaneously. However, this could cause the BIOS to be starved out of the
simultaneously. However, this could cause the BIOS to be starved out of the
lock - especially in cases such as the Embedded Controller driver where
there is a tight coupling between the OS and the BIOS.

Implemented an optimization for the ACPI Global Lock interrupt mechanism.
The Global Lock interrupt handler no longer queues the execution of a
separate thread to signal the global lock semaphore. Instead, the semaphore
separate thread to signal the global lock semaphore. Instead, the semaphore
is signaled directly from the interrupt handler.

Implemented support within the AML interpreter for package objects that
contain a larger AML length (package list length) than the package element
contain a larger AML length (package list length) than the package element
count. In this case, the length of the package is truncated to match the
package element count. Some BIOS code apparently modifies the package length
on the fly, and this change supports this behavior. Provides compatibility
package element count. Some BIOS code apparently modifies the package length
on the fly, and this change supports this behavior. Provides compatibility
with the MS AML interpreter. (With assistance from Fiodor Suietov)

Implemented a temporary fix for the BankValue parameter of a Bank Field to
Implemented a temporary fix for the BankValue parameter of a Bank Field to
support all constant values, now including the Zero and One opcodes.

Evaluation of this parameter must eventually be converted to a full TermArg evaluation. A not-implemented error is now returned (temporarily) for non-
Evaluation of this parameter must eventually be converted to a full TermArg evaluation. A not-implemented error is now returned (temporarily) for non-
constant values for this parameter.

Fixed problem reports (Fiodor Suietov) integrated:
- Fix for premature object deletion after CopyObject on Operation Region (BZ
- Fix for premature object deletion after CopyObject on Operation Region (BZ
350)

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has**
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.7K Code, 18.0K Data,  98.7K Total
    Debug Version:     160.9K Code, 65.1K Data, 226.0K Total
  Current Release:
    Non-Debug Version:  80.7K Code, 17.9K Data,  98.6K Total
    Debug Version:     161.0K Code, 65.1K Data, 226.1K Total
```

2) iASL Compiler/Disassembler and Tools:

No changes for this release.

---

07 July 2006. Summary of changes for version 20060707:

1) ACPI CA Core Subsystem:

Added the ACPI_PACKED_POINTERS_NOT_SUPPORTED macro to support C compilers that do not allow the initialization of address pointers within packed structures - even though the hardware itself may support misaligned transfers. Some of the debug data structures are packed by default to minimize size.

Added an error message for the case where AcpiOsGetThreadId() returns zero.
Added an error message for the case where AcpiOsGetThreadId() returns zero.
A non-zero value is required by the core ACPICA code to ensure the proper operation of AML mutexes and recursive control methods.

The DSDT is now the only ACPI table that determines whether the AML **interpreter is in 32-bit or 64-bit mode. Not really a functional change, but the hooks for per-table 32/64 switching have been removed from the code. A**
interpreter is in 32-bit or 64-bit mode. Not really a functional change, but the hooks for per-table 32/64 switching have been removed from the code. A
clarification to the ACPI specification is forthcoming in ACPI 3.0B.

Fixed a possible leak of an OwnerID in the error path of AcpiTbInitTableDescriptor (tbinstal.c), and migrated all table OwnerID **deletion to a single place in AcpiTbUninstallTable to correct possible leaks**
deletion to a single place in AcpiTbUninstallTable to correct possible leaks
when using the AcpiTbDeleteTablesByType interface (with assistance from Lance Ortiz.)

Fixed a problem with Serialized control methods where the semaphore associated with the method could be over-signaled after multiple method invocations.

Fixed two issues with the locking of the internal namespace data structure.
Fixed two issues with the locking of the internal namespace data structure.
Both the Unload() operator and AcpiUnloadTable interface now lock the namespace during the namespace deletion associated with the table unload (with assistance from Linn Crosetto.)

Fixed problem reports (Valery Podrezov) integrated:
- Eliminate unnecessary memory allocation for CreateXxxxField (BZ 5426)

Fixed problem reports (Fiodor Suietov) integrated:
- Incomplete cleanup branches in AcpiTbGetTableRsdt (BZ 369)
- On Address Space handler deletion, needless deactivation call (BZ 374)
- AcpiRemoveAddressSpaceHandler: validate Device handle parameter (BZ 375)
- Possible memory leak, Notify sub-objects of Processor, Power, ThermalZone
- AcpiRemoveAddressSpaceHandler: validate Device handle parameter (BZ 375)
- Possible memory leak, Notify sub-objects of Processor, Power, ThermalZone
(BZ 376)
- AcpiRemoveAddressSpaceHandler: validate Handler parameter (BZ 378)
- Minimum Length of RSDT should be validated (BZ 379)
- AcpiRemoveNotifyHandler: return AE_NOT_EXIST if Processor Obj has no Handler (BZ (380)
- AcpiUnloadTable: return AE_NOT_EXIST if no table of specified type loaded
- AcpiUnloadTable: return AE_NOT_EXIST if no table of specified type loaded
(BZ 381)

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has**
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.5K Code, 17.8K Data,  98.3K Total
    Debug Version:     160.8K Code, 64.8K Data, 225.6K Total
  Current Release:
    Non-Debug Version:  80.7K Code, 17.9K Data,  98.6K Total
    Debug Version:     161.0K Code, 65.1K Data, 226.1K Total
```

2) iASL Compiler/Disassembler and Tools:

Fixed problem reports:
Compiler segfault when ASL contains a long (>1024) String declaration (BZ 436)

---

23 June 2006. Summary of changes for version 20060623:

1) ACPI CA Core Subsystem:

Implemented a new ACPI_SPINLOCK type for the OSL lock interfaces. This allows the type to be customized to the host OS for improved efficiency (since a spinlock is usually a very small object.)

Implemented support for "ignored" bits in the ACPI registers. According to

Implemented support for "ignored" bits in the ACPI registers. According to
the ACPI specification, these bits should be preserved when writing the
registers via a read/modify/write cycle. There are 3 bits preserved in
this
registers via a read/modify/write cycle. There are 3 bits preserved in this
manner: PM1_CONTROL[0] (SCI_EN), PM1_CONTROL[9], and PM1_STATUS[11].

Implemented the initial deployment of new OSL mutex interfaces. Since
some
Implemented the initial deployment of new OSL mutex interfaces. Since some
host operating systems have separate mutex and semaphore objects, this
feature was requested. The base code now uses mutexes (and the new mutex
interfaces) wherever a binary semaphore was used previously. However, for
the current release, the mutex interfaces are defined as macros to map
them
to the existing semaphore interfaces. Therefore, no OSL changes are
required
the current release, the mutex interfaces are defined as macros to map them
to the existing semaphore interfaces. Therefore, no OSL changes are required
at this time. (See acpiosxf.h)

Fixed several problems with the support for the control method SyncLevel
parameter. The SyncLevel now works according to the ACPI specification
and
in concert with the Mutex SyncLevel parameter, since the current
SyncLevel
is a property of the executing thread. Mutual exclusion for control
methods
parameter. The SyncLevel now works according to the ACPI specification and
in concert with the Mutex SyncLevel parameter, since the current SyncLevel
is a property of the executing thread. Mutual exclusion for control methods
is now implemented with a mutex instead of a semaphore.

Fixed three instances of the use of the C shift operator in the bitfield
support code (exfldio.c) to avoid the use of a shift value larger than
the
target data width. The behavior of C compilers is undefined in this case
and
can cause unpredictable results, and therefore the case must be detected
and
support code (exfldio.c) to avoid the use of a shift value larger than the
target data width. The behavior of C compilers is undefined in this case and
can cause unpredictable results, and therefore the case must be detected and
avoided. (Fiodor Suietov)

Added an info message whenever an SSDT or OEM table is loaded dynamically
via the Load() or LoadTable() ASL operators. This should improve
debugging
capability since it will show exactly what tables have been loaded
(beyond
via the Load() or LoadTable() ASL operators. This should improve debugging
capability since it will show exactly what tables have been loaded (beyond
the tables present in the RSDT/XSDT.)

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.0K Code, 17.6K Data,  97.6K Total
    Debug Version:     160.2K Code, 64.7K Data, 224.9K Total
  Current Release:
    Non-Debug Version:  80.5K Code, 17.8K Data,  98.3K Total
```



```
    Debug Version:     160.8K Code, 64.8K Data, 225.6K Total
```

2) iASL Compiler/Disassembler and Tools:

No changes for this release.

---

08 June 2006. Summary of changes for version 20060608:

1) ACPI CA Core Subsystem:

Converted the locking mutex used for the ACPI hardware to a spinlock.
This
Converted the locking mutex used for the ACPI hardware to a spinlock. This
change should eliminate all problems caused by attempting to acquire a
semaphore at interrupt level, and it means that all ACPICA external
interfaces that directly access the ACPI hardware can be safely called
from
interrupt level. OSL code that implements the semaphore interfaces should
be
interfaces that directly access the ACPI hardware can be safely called from
interrupt level. OSL code that implements the semaphore interfaces should be
able to eliminate any workarounds for being called at interrupt level.

Fixed a regression introduced in 20060526 where the ACPI device
initialization could be prematurely aborted with an AE_NOT_FOUND if a
device
initialization could be prematurely aborted with an AE_NOT_FOUND if a device
did not have an optional _INI method.

Fixed an IndexField issue where a write to the Data Register should be
limited in size to the AccessSize (width) of the IndexField itself. (BZ
433,
limited in size to the AccessSize (width) of the IndexField itself. (BZ 433,
Fiodor Suietov)

Fixed problem reports (Valery Podrezov) integrated:
- Allow store of ThermalZone objects to Debug object (BZ 5369/5370)

Fixed problem reports (Fiodor Suietov) integrated:
- AcpiGetTableHeader doesn’t handle multiple instances correctly (BZ 364)

Removed four global mutexes that were obsolete and were no longer being
used.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.0K Code, 17.7K Data,  97.7K Total
    Debug Version:     160.3K Code, 64.9K Data, 225.2K Total
  Current Release:
    Non-Debug Version:  80.0K Code, 17.6K Data,  97.6K Total
    Debug Version:     160.2K Code, 64.7K Data, 224.9K Total
```

2) iASL Compiler/Disassembler and Tools:

Fixed a fault when using -g option (get tables from registry) on Windows
machines.

Fixed problem reports integrated:
- Generate error if CreateField NumBits parameter is zero. (BZ 405)
- Fault if Offset/Length in Field unit is very large (BZ 432, Fiodor Suietov)
- Global table revision override (-r) is ignored (BZ 413)

---
26 May 2006. Summary of changes for version 20060526:

1) ACPI CA Core Subsystem:

Restructured, flattened, and simplified the internal interfaces for **namespace object evaluation - resulting in smaller code, less CPU stack use,** *namespace object evaluation - resulting in smaller code, less CPU stack use,* and fewer interfaces. (With assistance from Mikhail Kouzmich)

Fixed a problem with the CopyObject operator where the first parameter was not typed correctly for the parser, interpreter, compiler, and disassembler. *Fixed a problem with the CopyObject operator where the first parameter was not typed correctly for the parser, interpreter, compiler, and disassembler.* Caused various errors and unexpected behavior.

Fixed a problem where a ShiftLeft or ShiftRight of more than 64 bits produced incorrect results with some C compilers. Since the behavior of C compilers when the shift value is larger than the datatype width is **apparently not well defined, the interpreter now detects this condition and** *apparently not well defined, the interpreter now detects this condition and* simply returns zero as expected in all such cases. (BZ 395)

Fixed problem reports (Valery Podrezov) integrated:
- Update String-to-Integer conversion to match ACPI 3.0A spec (BZ 5329)
- Allow interpreter to handle nested method declarations (BZ 5361)

Fixed problem reports (Fiodor Suietov) integrated:
- AcpiTerminate doesn’t free debug memory allocation list objects (BZ 355)
- After Core Subsystem shutdown, AcpiSubsystemStatus returns AE_OK (BZ 356)
- AcpiTerminate doesn’t free debug memory allocation list objects (BZ 355)
- After Core Subsystem shutdown, AcpiSubsystemStatus returns AE_OK (BZ 356)
- AcpiOsUnmapMemory for RSDP can be invoked inconsistently (BZ 357)
- Resource Manager should return AE_TYPE for non-device objects (BZ 358)
- Incomplete cleanup branch in AcpiNsEvaluateRelative (BZ 359)
- Use AcpiOsFree instead of ACPI_FREE in AcpiRsSetSrsMethodData (BZ 360)
- Incomplete cleanup branch in AcpiPsParseAml (BZ 361)
- Incomplete cleanup branch in AcpiDsDeleteWalkState (BZ 362)
- AcpiGetTableHeader returns AE_NO_ACPI_TABLES until DSDT is loaded (BZ 365)
- AcpiGetTableHeader returns AE_NO_ACPI_TABLES until DSDT is loaded (BZ 365)
- Status of the Global Initialization Handler call not used (BZ 366)
- Incorrect object parameter to Global Initialization Handler (BZ 367)

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has** *debug version of the code includes the debug output trace mechanism and has* a much larger code and data size.

  Previous Release:
    Non-Debug Version: 79.8K Code, 17.7K Data, 97.5K Total
    Debug Version: 160.5K Code, 65.1K Data, 225.6K Total

  Current Release:
    Non-Debug Version: 80.0K Code, 17.7K Data, 97.7K Total
    Debug Version: 160.3K Code, 64.9K Data, 225.2K Total

2) iASL Compiler/Disassembler and Tools:

Modified the parser to allow the names IO, DMA, and IRQ to be used as namespace identifiers with no collision with existing resource descriptor macro names. This provides compatibility with other ASL compilers and is **most useful for disassembly/recompilation of existing tables without parse** *most useful for disassembly/recompilation of existing tables without parse* errors. (With assistance from Thomas Renninger)

Disassembler: fixed an incorrect disassembly problem with the DataTableRegion and CopyObject operators. Fixed a possible fault during disassembly of some Alias operators.

---
12 May 2006. Summary of changes for version 20060512:

1) ACPI CA Core Subsystem:

Replaced the AcpiOsQueueForExecution interface with a new interface named **AcpiOsExecute. The major difference is that the new interface does not have a Priority parameter, this appeared to be useless and has been replaced by a** *AcpiOsExecute. The major difference is that the new interface does not have a Priority parameter, this appeared to be useless and has been replaced by a* Type parameter. The Type tells the host what type of execution is being requested, such as global lock handler, notify handler, GPE handler, etc. **This allows the host to queue and execute the request as appropriate for the request type, possibly using different work queues and different priorities** *This allows the host to queue and execute the request as appropriate for the request type, possibly using different work queues and different priorities* for the various request types. This enables fixes for multithreading **deadlock problems such as BZ #5534, and will require changes to all existing** *deadlock problems such as BZ #5534, and will require changes to all existing* OS interface layers. (Alexey Starikovskiy and Bob Moore)

Fixed a possible memory leak associated with the support for the so-called *Fixed a possible memory leak associated with the support for the so-called* "implicit return" ACPI extension. Reported by FreeBSD, BZ #6514. (Fiodor Suietov)

Fixed a problem with the Load() operator where a table load from an **operation region could overwrite an internal table buffer by up to 7 bytes and cause alignment faults on IPF systems. (With assistance from Luming Yu)** *operation region could overwrite an internal table buffer by up to 7 bytes and cause alignment faults on IPF systems. (With assistance from Luming Yu)*

Example Code and Data Size: These are the sizes for the OS-independent acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The **debug version of the code includes the debug output trace mechanism and has** *debug version of the code includes the debug output trace mechanism and has* a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  79.7K Code, 17.7K Data,  97.4K Total
    Debug Version:     160.1K Code, 65.2K Data, 225.3K Total
  Current Release:
    Non-Debug Version:  79.8K Code, 17.7K Data,  97.5K Total
    Debug Version:     160.5K Code, 65.1K Data, 225.6K Total


2) iASL Compiler/Disassembler and Tools:

Disassembler: Implemented support to cross reference the internal
namespace
and automatically generate ASL External() statements for symbols not
defined
Disassembler: Implemented support to cross reference the internal namespace
and automatically generate ASL External() statements for symbols not defined
within the current table being disassembled. This will simplify the
disassembly and recompilation of interdependent tables such as SSDTs
since
disassembly and recompilation of interdependent tables such as SSDTs since
these statements will no longer have to be added manually.

Disassembler: Implemented experimental support to automatically detect
invocations of external control methods and generate appropriate
External()
statements. This is problematic because the AML cannot be correctly
parsed
until the number of arguments for each control method is known.
Currently,
invocations of external control methods and generate appropriate External()
statements. This is problematic because the AML cannot be correctly parsed
until the number of arguments for each control method is known. Currently,
standalone method invocations and invocations as the source operand of a
Store() statement are supported.

Disassembler: Implemented support for the ASL pseudo-operators LNotEqual,
LLessEqual, and LGreaterEqual. Previously disassembled as LNot(LEqual()),
LNot(LGreater()), and LNot(LLess()), this makes the disassembled ASL code
more readable and likely closer to the original ASL source.

----------------------------------------
21 April 2006. Summary of changes for version 20060421:

1) ACPI CA Core Subsystem:

Removed a device initialization optimization introduced in 20051216 where
the _STA method was not run unless an _INI was also present for the same
device. This optimization could cause problems because it could allow
_INI
device. This optimization could cause problems because it could allow _INI
methods to be run within a not-present device subtree. (If a not-present
device had no _INI, _STA would not be run, the not-present status would
not
device had no _INI, _STA would not be run, the not-present status would not
be discovered, and the children of the device would be incorrectly
traversed.)

Implemented a new _STA optimization where namespace subtrees that do not
contain _INI are identified and ignored during device initialization.
Selectively running _STA can significantly improve boot time on large
machines (with assistance from Len Brown.)

Implemented support for the device initialization case where the returned
_STA flags indicate a device not-present but functioning. In this case,
```



```
_INI
_STA flags indicate a device not-present but functioning. In this case, _INI
is not run, but the device children are examined for presence, as per the
ACPI specification.

Implemented an additional change to the IndexField support in order to
conform to MS behavior. The value written to the Index Register is not
simply a byte offset, it is a byte offset in units of the access width of
the parent Index Field. (Fiodor Suietov)

Defined and deployed a new OSL interface, AcpiOsValidateAddress. This
interface is called during the creation of all AML operation regions, and
allows the host OS to exert control over what addresses it will allow the
AML code to access. Operation Regions whose addresses are disallowed will
cause a runtime exception when they are actually accessed (will not
affect
cause a runtime exception when they are actually accessed (will not affect
or abort table loading.) See oswinxf or osunixxf for an example
implementation.

Defined and deployed a new OSL interface, AcpiOsValidateInterface. This
interface allows the host OS to match the various "optional"
interface/behavior strings for the _OSI predefined control method as
appropriate (with assistance from Bjorn Helgaas.) See oswinxf or osunixxf
for an example implementation.

Restructured and corrected various problems in the exception handling
code
Restructured and corrected various problems in the exception handling code
paths within DsCallControlMethod and DsTerminateControlMethod in dsmethod
(with assistance from Takayoshi Kochi.)

Modified the Linux source converter to ignore quoted string literals
while
converting identifiers from mixed to lower case. This will correct
problems
Modified the Linux source converter to ignore quoted string literals while
converting identifiers from mixed to lower case. This will correct problems
with the disassembler and other areas where such strings must not be
modified.

The ACPI_FUNCTION_* macros no longer require quotes around the function
name. This allows the Linux source converter to convert the names, now
that
name. This allows the Linux source converter to convert the names, now that
the converter ignores quoted strings.

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and
has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

  Previous Release:

    Non-Debug Version:  81.1K Code, 17.7K Data,  98.8K Total
    Debug Version:     158.9K Code, 64.9K Data, 223.8K Total
  Current Release:
    Non-Debug Version:  79.7K Code, 17.7K Data,  97.4K Total
    Debug Version:     160.1K Code, 65.2K Data, 225.3K Total


2) iASL Compiler/Disassembler and Tools:

Implemented 3 new warnings for iASL, and implemented multiple warning
```

levels
Implemented 3 new warnings for iASL, and implemented multiple warning levels
(w2 flag).

1) Ignored timeouts: If the TimeoutValue parameter to Wait or Acquire is not
1) Ignored timeouts: If the TimeoutValue parameter to Wait or Acquire is not
WAIT_FOREVER (0xFFFF) and the code does not examine the return value to
check for the possible timeout, a warning is issued.

2) Useless operators: If an ASL operator does not specify an optional target
2) Useless operators: If an ASL operator does not specify an optional target
operand and it also does not use the function return value from the
operator, a warning is issued since the operator effectively does nothing.
operator, a warning is issued since the operator effectively does nothing.

3) Unreferenced objects: If a namespace object is created, but never
referenced, a warning is issued. This is a warning level 2 since there are
cases where this is ok, such as when a secondary table is loaded that uses
the unreferenced objects. Even so, care is taken to only flag objects that
referenced, a warning is issued. This is a warning level 2 since there are
cases where this is ok, such as when a secondary table is loaded that uses
the unreferenced objects. Even so, care is taken to only flag objects that
don’t look like they will ever be used. For example, the reserved methods
(starting with an underscore) are usually not referenced because it is
expected that the OS will invoke them.

---
31 March 2006. Summary of changes for version 20060331:

1) ACPI CA Core Subsystem:

Implemented header file support for the following additional ACPI tables:
ASF!, BOOT, CPEP, DBGP, MCFG, SPCR, SPMI, TCPA, and WDRT. With this support, all current and known ACPI tables are now defined in the ACPICA headers and
ASF!, BOOT, CPEP, DBGP, MCFG, SPCR, SPMI, TCPA, and WDRT. With this support,
all current and known ACPI tables are now defined in the ACPICA headers and
are available for use by device drivers and other software.

Implemented support to allow tables that contain ACPI names with invalid
characters to be loaded. Previously, this would cause the table load to
fail, but since there are several known cases of such tables on existing
machines, this change was made to enable ACPI support for them. Also, this
machines, this change was made to enable ACPI support for them. Also, this
matches the behavior of the Microsoft ACPI implementation.

Fixed a couple regressions introduced during the memory optimization in the
Fixed a couple regressions introduced during the memory optimization in the
20060317 release. The namespace node definition required additional
reorganization and an internal datatype that had been changed to 8-bit was
reorganization and an internal datatype that had been changed to 8-bit was
restored to 32-bit. (Valery Podrezov)

Fixed a problem where a null pointer passed to AcpiUtDeleteGenericState
could be passed through to AcpiOsReleaseObject which is unexpected. Such
null pointers are now trapped and ignored, matching the behavior of the

previous implementation before the deployment of AcpiOsReleaseObject.
(Valery Podrezov, Fiodor Suietov)

Fixed a memory mapping leak during the deletion of a SystemMemory operation
Fixed a memory mapping leak during the deletion of a SystemMemory operation
region where a cached memory mapping was not deleted. This became a
noticeable problem for operation regions that are defined within frequently
noticeable problem for operation regions that are defined within frequently
used control methods. (Dana Meyers)

Reorganized the ACPI table header files into two main files: one for the
ACPI tables consumed by the ACPICA core, and another for the miscellaneous
ACPI tables that are consumed by the drivers and other software. The various
ACPI tables consumed by the ACPICA core, and another for the miscellaneous
ACPI tables that are consumed by the drivers and other software. The various
FADT definitions were merged into one common section and three different
tables (ACPI 1.0, 1.0+, and 2.0)

Example Code and Data Size: These are the sizes for the OS-independent
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
debug version of the code includes the debug output trace mechanism and has
debug version of the code includes the debug output trace mechanism and has
a much larger code and data size.

```
  Previous Release:
    Non-Debug Version:  80.9K Code, 17.7K Data,  98.6K Total
    Debug Version:     158.7K Code, 64.8K Data, 223.5K Total
  Current Release:
    Non-Debug Version:  81.1K Code, 17.7K Data,  98.8K Total
    Debug Version:     158.9K Code, 64.9K Data, 223.8K Total
```

2) iASL Compiler/Disassembler and Tools:

Disassembler: Implemented support to decode and format all non-AML ACPI
tables (tables other than DSDTs and SSDTs.) This includes the new tables
added to the ACPICA headers, therefore all current and known ACPI tables are
added to the ACPICA headers, therefore all current and known ACPI tables are
supported.

Disassembler: The change to allow ACPI names with invalid characters also
enables the disassembly of such tables. Invalid characters within names are
enables the disassembly of such tables. Invalid characters within names are
changed to ’*’ to make the name printable; the iASL compiler will still
generate an error for such names, however, since this is an invalid ACPI
character.

Implemented an option for AcpiXtract (-a) to extract all tables found in the
Implemented an option for AcpiXtract (-a) to extract all tables found in the
input file. The default invocation extracts only the DSDTs and SSDTs.

Fixed a couple of gcc generation issues for iASL and AcpiExec and added a
makefile for the AcpiXtract utility.

---
17 March 2006. Summary of changes for version 20060317:

1) ACPI CA Core Subsystem:

Implemented the use of a cache object for all internal namespace nodes. Since there are about 1000 static nodes in a typical system, this will **decrease memory use for cache implementations that minimize per-allocation**
decrease memory use for cache implementations that minimize per-allocation overhead (such as a slab allocator.)

Removed the reference count mechanism for internal namespace nodes, since it
Removed the reference count mechanism for internal namespace nodes, since it was deemed unnecessary. This reduces the size of each namespace node by **about 5%-10% on all platforms. Nodes are now 20 bytes for the 32-bit case,**
about 5%-10% on all platforms. Nodes are now 20 bytes for the 32-bit case, and 32 bytes for the 64-bit case.

Optimized several internal data structures to reduce object size on 64-bit
Optimized several internal data structures to reduce object size on 64-bit platforms by packing data within the 64-bit alignment. This includes the frequently used ACPI_OPERAND_OBJECT, of which there can be ~1000 static instances corresponding to the namespace objects.

Added two new strings for the predefined _OSI method: "Windows 2001.1 SP1"
Added two new strings for the predefined _OSI method: "Windows 2001.1 SP1" and "Windows 2006".

Split the allocation tracking mechanism out to a separate file, from utalloc.c to uttrack.c. This mechanism appears to be only useful for application-level code. Kernels may wish to not include uttrack.c in distributions.

Removed all remnants of the obsolete ACPI_REPORT_* macros and the associated
Removed all remnants of the obsolete ACPI_REPORT_* macros and the associated code. (These macros have been replaced by the ACPI_ERROR and ACPI_WARNING macros.)

Code and Data Size: These are the sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output
Code and Data Size: These are the sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  81.1K Code, 17.8K Data,  98.9K Total
    Debug Version:     161.6K Code, 65.7K Data, 227.3K Total
  Current Release:
    Non-Debug Version:  80.9K Code, 17.7K Data,  98.6K Total
    Debug Version:     158.7K Code, 64.8K Data, 223.5K Total
```

2) iASL Compiler/Disassembler and Tools:

Implemented an ANSI C version of the acpixtract utility. This version will
Implemented an ANSI C version of the acpixtract utility. This version will automatically extract the DSDT and all SSDTs from the input acpidump text file and dump the binary output to separate files. It can also display a summary of the input file including the headers for each table found and will extract any single ACPI table, with any signature. (See source/tools/acpixtract)

---

10 March 2006. Summary of changes for version 20060310:

1) ACPI CA Core Subsystem:

Tagged all external interfaces to the subsystem with the new **ACPI_EXPORT_SYMBOL macro. This macro can be defined as necessary to assist**
ACPI_EXPORT_SYMBOL macro. This macro can be defined as necessary to assist kernel integration. For Linux, the macro resolves to the EXPORT_SYMBOL macro. The default definition is NULL.

Added the ACPI_THREAD_ID type for the return value from AcpiOsGetThreadId.
Added the ACPI_THREAD_ID type for the return value from AcpiOsGetThreadId. This allows the host to define this as necessary to simplify kernel integration. The default definition is ACPI_NATIVE_UINT.

Fixed two interpreter problems related to error processing, the deletion of
Fixed two interpreter problems related to error processing, the deletion of objects, and placing invalid pointers onto the internal operator result stack. BZ 6028, 6151 (Valery Podrezov)

Increased the reference count threshold where a warning is emitted for large reference counts in order to eliminate unnecessary warnings on systems with
Increased the reference count threshold where a warning is emitted for large reference counts in order to eliminate unnecessary warnings on systems with large namespaces (especially 64-bit.) Increased the value from 0x400 to 0x800.

Due to universal disagreement as to the meaning of the 'c' in the calloc()
Due to universal disagreement as to the meaning of the 'c' in the calloc() function, the ACPI_MEM_CALLOCATE macro has been renamed to ACPI_ALLOCATE_ZEROED so that the purpose of the interface is 'clear'. ACPI_MEM_ALLOCATE and ACPI_MEM_FREE are renamed to ACPI_ALLOCATE and ACPI_FREE.

Code and Data Size: These are the sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output
Code and Data Size: These are the sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  81.0K Code, 17.8K Data,  98.8K Total
    Debug Version:     161.4K Code, 65.7K Data, 227.1K Total
  Current Release:
    Non-Debug Version:  81.1K Code, 17.8K Data,  98.9K Total
    Debug Version:     161.6K Code, 65.7K Data, 227.3K Total
```

2) iASL Compiler/Disassembler:

Disassembler: implemented support for symbolic resource descriptor references. If a CreateXxxxField operator references a fixed offset within a resource descriptor, a name is assigned to the descriptor and the offset is translated to the appropriate resource tag and pathname. The addition of this support brings the disassembled code very close to the original ASL source code and helps eliminate run-time errors when the disassembled code is modified (and recompiled) in such a way as to invalidate the original fixed offsets.

Implemented support for a Descriptor Name as the last parameter to the ASL Register() macro. This parameter was inadvertently left out of the ACPI specification, and will be added for ACPI 3.0b.

Fixed a problem where the use of the "_OSI" string (versus the full path "_OSI") caused an internal compiler error. ("No back ptr to op")

Fixed a problem with the error message that occurs when an invalid string is used for a _HID object (such as one with an embedded asterisk: "*PNP010A".) The correct message is now displayed.

---

17 February 2006. Summary of changes for version 20060217:

1) ACPI CA Core Subsystem:

Implemented a change to the IndexField support to match the behavior of the Microsoft AML interpreter. The value written to the Index register is now a byte offset, no longer an index based upon the width of the Data register. This should fix IndexField problems seen on some machines where the Data register is not exactly one byte wide. The ACPI specification will be clarified on this point.

Fixed a problem where several resource descriptor types could overrun the internal descriptor buffer due to size miscalculation: VendorShort, VendorLong, and Interrupt. This was noticed on IA64 machines, but could affect all platforms.

Fixed a problem where individual resource descriptors were misaligned within the internal buffer, causing alignment faults on IA64 platforms.

Code and Data Size: These are the sizes for the acpica.lib produced by

the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output

Code and Data Size: These are the sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output

trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  81.1K Code, 17.8K Data,  98.9K Total
    Debug Version:     161.3K Code, 65.6K Data, 226.9K Total
  Current Release:
    Non-Debug Version:  81.0K Code, 17.8K Data,  98.8K Total
    Debug Version:     161.4K Code, 65.7K Data, 227.1K Total
```

2) iASL Compiler/Disassembler:

Implemented support for new reserved names: _WDG and _WED are Microsoft extensions for Windows Instrumentation Management, _TDL is a new ACPI-defined method (Throttling Depth Limit.)

Fixed a problem where a zero-length VendorShort or VendorLong resource descriptor was incorrectly emitted as a descriptor of length one.

---

10 February 2006. Summary of changes for version 20060210:

1) ACPI CA Core Subsystem:

Removed a couple of extraneous ACPI_ERROR messages that appeared during normal execution. These became apparent after the conversion from ACPI_DEBUG_PRINT.

Fixed a problem where the CreateField operator could hang if the BitIndex or NumBits parameter referred to a named object. (Valery Podrezov, BZ 5359)

Fixed a problem where a DeRefOf operation on a buffer object incorrectly failed with an exception. This also fixes a couple of related RefOf and DeRefOf issues. (Valery Podrezov, BZ 5360/5392/5387)

Fixed a problem where the AE_BUFFER_LIMIT exception was returned instead of AE_STRING_LIMIT on an out-of-bounds Index() operation. (Valery Podrezov, BZ 5480)

Implemented a memory cleanup at the end of the execution of each iteration of an AML While() loop, preventing the accumulation of outstanding objects. (Valery Podrezov, BZ 5427)

Eliminated a chunk of duplicate code in the object resolution code. (Valery

Podrezov, BZ 5336)

Fixed several warnings during the 64-bit code generation.

The AcpiSrc source code conversion tool now inserts one line of whitespace after an if() statement that is followed immediately by a comment, improving
The AcpiSrc source code conversion tool now inserts one line of whitespace after an if() statement that is followed immediately by a comment, improving readability of the Linux code.

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the**
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency**
and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  81.0K Code, 17.9K Data,  98.9K Total
    Debug Version:     161.3K Code, 65.7K Data, 227.0K Total
  Current Release:
    Non-Debug Version:  81.1K Code, 17.8K Data,  98.9K Total
    Debug Version:     161.3K Code, 65.6K Data, 226.9K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem with the disassembly of a BankField operator with a complex
Fixed a problem with the disassembly of a BankField operator with a complex expression for the BankValue parameter.

---
27 January 2006. Summary of changes for version 20060127:

1) ACPI CA Core Subsystem:

Implemented support in the Resource Manager to allow unresolved namestring references within resource package objects for the _PRT method. This support
Implemented support in the Resource Manager to allow unresolved namestring references within resource package objects for the _PRT method. This support is in addition to the previously implemented unresolved reference support within the AML parser. If the interpreter slack mode is enabled, these **unresolved references will be passed through to the caller as a NULL package**
unresolved references will be passed through to the caller as a NULL package entry.

Implemented and deployed new macros and functions for error and warning messages across the subsystem. These macros are simpler and generate less code than their predecessors. The new macros ACPI_ERROR, ACPI_EXCEPTION, ACPI_WARNING, and ACPI_INFO replace the ACPI_REPORT_* macros. The older macros remain defined to allow ACPI drivers time to migrate to the new macros.

Implemented the ACPI_CPU_FLAGS type to simplify host OS integration of the
Implemented the ACPI_CPU_FLAGS type to simplify host OS integration of the Acquire/Release Lock OSL interfaces.

Fixed a problem where Alias ASL operators are sometimes not correctly resolved, in both the interpreter and the iASL compiler.

Fixed several problems with the implementation of the ConcatenateResTemplate
Fixed several problems with the implementation of the ConcatenateResTemplate ASL operator. As per the ACPI specification, zero length buffers are now treated as a single EndTag. One-length buffers always cause a fatal **exception. Non-zero length buffers that do not end with a full 2-byte EndTag**
exception. Non-zero length buffers that do not end with a full 2-byte EndTag cause a fatal exception.

Fixed a possible structure overwrite in the AcpiGetObjectInfo external interface. (With assistance from Thomas Renninger)

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the**
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency**
and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  83.1K Code, 18.4K Data, 101.5K Total
    Debug Version:     163.2K Code, 66.2K Data, 229.4K Total
  Current Release:
    Non-Debug Version:  81.0K Code, 17.9K Data,  98.9K Total
    Debug Version:     161.3K Code, 65.7K Data, 227.0K Total
```

2) iASL Compiler/Disassembler:

Fixed an internal error that was generated for any forward references to ASL
Fixed an internal error that was generated for any forward references to ASL Alias objects.

---
13 January 2006. Summary of changes for version 20060113:

1) ACPI CA Core Subsystem:

Added 2006 copyright to all module headers and signons. This affects virtually every file in the ACPICA core subsystem, iASL compiler, and the utilities.

Enhanced the ACPICA error reporting in order to simplify user migration to
Enhanced the ACPICA error reporting in order to simplify user migration to the non-debug version of ACPICA. Replaced all instances of the **ACPI_DEBUG_PRINT macro invoked at the ACPI_DB_ERROR and ACPI_DB_WARN debug**
ACPI_DEBUG_PRINT macro invoked at the ACPI_DB_ERROR and ACPI_DB_WARN debug

levels with the ACPI_REPORT_ERROR and ACPI_REPORT_WARNING macros,
respectively. This preserves all error and warning messages in the non-
debug
respectively. This preserves all error and warning messages in the non-debug
version of the ACPICA code (this has been referred to as the "debug lite"
option.) Over 200 cases were converted to create a total of over 380
error/warning messages across the ACPICA code. This increases the code
and
data size of the default non-debug version of the code somewhat (about
13K),
error/warning messages across the ACPICA code. This increases the code and
data size of the default non-debug version of the code somewhat (about 13K),
but all error/warning reporting may be disabled if desired (and code
eliminated) by specifying the ACPI_NO_ERROR_MESSAGES compile-time
configuration option. The size of the debug version of ACPICA remains
about
configuration option. The size of the debug version of ACPICA remains about
the same.

Fixed a memory leak within the AML Debugger "Set" command. One object was
not properly deleted for every successful invocation of the command.

Code and Data Size: The current and previous library sizes for the core
subsystem are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.
These
values do not include any ACPI driver or OSPM code. The debug version of
the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These
values do not include any ACPI driver or OSPM code. The debug version of the
code includes the debug output trace mechanism and has a much larger code
and data size. Note that these values will vary depending on the
efficiency
and data size. Note that these values will vary depending on the efficiency
of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  76.6K Code, 12.3K Data,  88.9K Total
    Debug Version:     163.7K Code, 67.5K Data, 231.2K Total
  Current Release:
    Non-Debug Version:  83.1K Code, 18.4K Data, 101.5K Total
    Debug Version:     163.2K Code, 66.2K Data, 229.4K Total
```

2) iASL Compiler/Disassembler:

The compiler now officially supports the ACPI 3.0a specification that was
released on December 30, 2005. (Specification is available at
www.acpi.info)
released on December 30, 2005. (Specification is available at www.acpi.info)

---
16 December 2005. Summary of changes for version 20051216:

1) ACPI CA Core Subsystem:

Implemented optional support to allow unresolved names within ASL Package
objects. A null object is inserted in the package when a named reference
cannot be located in the current namespace. Enabled via the interpreter
slack flag, this should eliminate AE_NOT_FOUND exceptions seen on
machines
slack flag, this should eliminate AE_NOT_FOUND exceptions seen on machines
that contain such code.

Implemented an optimization to the initialization sequence that can
improve

boot time. During ACPI device initialization, the _STA method is now run
if
and only if the _INI method exists. The _STA method is used to determine
if
the device is present; An _INI can only be run if _STA returns present,
but
Implemented an optimization to the initialization sequence that can improve
boot time. During ACPI device initialization, the _STA method is now run if
and only if the _INI method exists. The _STA method is used to determine if
the device is present; An _INI can only be run if _STA returns present, but
it is a waste of time to run the _STA method if the _INI does not exist.
(Prototype and assistance from Dong Wei)

Implemented use of the C99 uintptr_t for the pointer casting macros if it
is
available in the current compiler. Otherwise, the default (void *) cast
is
Implemented use of the C99 uintptr_t for the pointer casting macros if it is
*available in the current compiler. Otherwise, the default (void *) cast is*
used as before.

Fixed some possible memory leaks found within the execution path of the
Break, Continue, If, and CreateField operators. (Valery Podrezov)

Fixed a problem introduced in the 20051202 release where an exception is
generated during method execution if a control method attempts to declare
another method.

Moved resource descriptor string constants that are used by both the AML
disassembler and AML debugger to the common utilities directory so that
these components are independent.

Implemented support in the AcpiExec utility (-e switch) to globally
ignore
Implemented support in the AcpiExec utility (-e switch) to globally ignore
exceptions during control method execution (method is not aborted.)

Added the rsinfo.c source file to the AcpiExec makefile for Linux/Unix
generation.

Code and Data Size: The current and previous library sizes for the core
subsystem are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.
These
values do not include any ACPI driver or OSPM code. The debug version of
the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These
values do not include any ACPI driver or OSPM code. The debug version of the
code includes the debug output trace mechanism and has a much larger code
and data size. Note that these values will vary depending on the
efficiency
and data size. Note that these values will vary depending on the efficiency
of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  76.3K Code, 12.3K Data,  88.6K Total
    Debug Version:     163.2K Code, 67.4K Data, 230.6K Total
  Current Release:
    Non-Debug Version:  76.6K Code, 12.3K Data,  88.9K Total
    Debug Version:     163.7K Code, 67.5K Data, 231.2K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem where a CPU stack overflow fault could occur if a
recursive

Fixed a problem where a CPU stack overflow fault could occur if a recursive
method call was made from within a Return statement.

---
02 December 2005. Summary of changes for version 20051202:

1) ACPI CA Core Subsystem:

Modified the parsing of control methods to no longer create namespace
objects during the first pass of the parse. Objects are now created only
during the execute phase, at the moment the namespace creation operator
is
encountered in the AML (Name, OperationRegion, CreateByteField, etc.)
This
during the execute phase, at the moment the namespace creation operator is
encountered in the AML (Name, OperationRegion, CreateByteField, etc.) This
should eliminate ALREADY_EXISTS exceptions seen on some machines where
reentrant control methods are protected by an AML mutex. The mutex will
now
correctly block multiple threads from attempting to create the same
object
reentrant control methods are protected by an AML mutex. The mutex will now
correctly block multiple threads from attempting to create the same object
more than once.

Increased the number of available Owner Ids for namespace object tracking
from 32 to 255. This should eliminate the OWNER_ID_LIMIT exceptions seen
on
some machines with a large number of ACPI tables (either static or
dynamic).
from 32 to 255. This should eliminate the OWNER_ID_LIMIT exceptions seen on
some machines with a large number of ACPI tables (either static or dynamic).

Fixed a problem with the AcpiExec utility where a fault could occur when
the
Fixed a problem with the AcpiExec utility where a fault could occur when the
-b switch (batch mode) is used.

Enhanced the namespace dump routine to output the owner ID for each
namespace object.

Code and Data Size: The current and previous library sizes for the core
subsystem are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.
These
values do not include any ACPI driver or OSPM code. The debug version of
the
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These
values do not include any ACPI driver or OSPM code. The debug version of the
code includes the debug output trace mechanism and has a much larger code
and data size. Note that these values will vary depending on the
efficiency
and data size. Note that these values will vary depending on the efficiency
of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  76.3K Code, 12.3K Data,  88.6K Total
    Debug Version:     163.0K Code, 67.4K Data, 230.4K Total
  Current Release:
    Non-Debug Version:  76.3K Code, 12.3K Data,  88.6K Total
    Debug Version:     163.2K Code, 67.4K Data, 230.6K Total
```

2) iASL Compiler/Disassembler:

Fixed a parse error during compilation of certain Switch/Case constructs.

To
simplify the parse, the grammar now allows for multiple Default
statements
Fixed a parse error during compilation of certain Switch/Case constructs. To
simplify the parse, the grammar now allows for multiple Default statements
and this error is now detected and flagged during the analysis phase.

Disassembler: The disassembly now includes the contents of the original
table header within a comment at the start of the file. This includes the
name and version of the original ASL compiler.

---
17 November 2005. Summary of changes for version 20051117:

1) ACPI CA Core Subsystem:

Fixed a problem in the AML parser where the method thread count could be
decremented below zero if any errors occurred during the method parse
phase.
This should eliminate AE_AML_METHOD_LIMIT exceptions seen on some
machines.
decremented below zero if any errors occurred during the method parse phase.
This should eliminate AE_AML_METHOD_LIMIT exceptions seen on some machines.
This also fixed a related regression with the mechanism that detects and
corrects methods that cannot properly handle reentrancy (related to the
deployment of the new OwnerId mechanism.)

Eliminated the pre-parsing of control methods (to detect errors) during
table load. Related to the problem above, this was causing unwind issues
if
any errors occurred during the parse, and it seemed to be overkill. A
table
table load. Related to the problem above, this was causing unwind issues if
any errors occurred during the parse, and it seemed to be overkill. A table
load should not be aborted if there are problems with any single control
method, thus rendering this feature rather pointless.

Fixed a problem with the new table-driven resource manager where an
internal
Fixed a problem with the new table-driven resource manager where an internal
buffer overflow could occur for small resource templates.

Implemented a new external interface, AcpiGetVendorResource. This
interface
will find and return a vendor-defined resource descriptor within a _CRS
or
_PRS method via an ACPI 3.0 UUID match. With assistance from Bjorn
Helgaas.
Implemented a new external interface, AcpiGetVendorResource. This interface
will find and return a vendor-defined resource descriptor within a _CRS or
_PRS method via an ACPI 3.0 UUID match. With assistance from Bjorn Helgaas.

Removed the length limit (200) on string objects as per the upcoming ACPI
3.0A specification. This affects the following areas of the interpreter:
1)
any implicit conversion of a Buffer to a String, 2) a String object
result
3.0A specification. This affects the following areas of the interpreter: 1)
any implicit conversion of a Buffer to a String, 2) a String object result
of the ASL Concatentate operator, 3) the String object result of the ASL
ToString operator.

Fixed a problem in the Windows OS interface layer (OSL) where a
WAIT_FOREVER
Fixed a problem in the Windows OS interface layer (OSL) where a WAIT_FOREVER
on a semaphore object would incorrectly timeout. This allows the

multithreading features of the AcpiExec utility to work properly under Windows.

Updated the Linux makefiles for the iASL compiler and AcpiExec to include the recently added file named "utresrc.c".

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the** *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the* code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency** *and data size. Note that these values will vary depending on the efficiency* of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  76.2K Code, 12.3K Data,  88.5K Total
    Debug Version:     163.0K Code, 67.4K Data, 230.4K Total
  Current Release:
    Non-Debug Version:  76.3K Code, 12.3K Data,  88.6K Total
    Debug Version:     163.0K Code, 67.4K Data, 230.4K Total
```

2) iASL Compiler/Disassembler:

Removed the limit (200) on string objects as per the upcoming ACPI 3.0A **specification. For the iASL compiler, this means that string literals within** *specification. For the iASL compiler, this means that string literals within* the source ASL can be of any length.

Enhanced the listing output to dump the AML code for resource descriptors **immediately after the ASL code for each descriptor, instead of in a block at** *immediately after the ASL code for each descriptor, instead of in a block at* the end of the entire resource template.

Enhanced the compiler debug output to dump the entire original parse tree **constructed during the parse phase, before any transforms are applied to the** *constructed during the parse phase, before any transforms are applied to the* tree. The transformed tree is dumped also.

---

02 November 2005. Summary of changes for version 20051102:

1) ACPI CA Core Subsystem:

Modified the subsystem initialization sequence to improve GPE support. The GPE initialization has been split into two parts in order to defer execution of the _PRW methods (Power Resources for Wake) until after the hardware is *Modified the subsystem initialization sequence to improve GPE support. The GPE initialization has been split into two parts in order to defer execution of the _PRW methods (Power Resources for Wake) until after the hardware is* fully initialized and the SCI handler is installed. This allows the _PRW **methods to access fields protected by the Global Lock. This will fix systems** *methods to access fields protected by the Global Lock. This will fix systems*

where a NO_GLOBAL_LOCK exception has been seen during initialization.

Converted the ACPI internal object disassemble and display code within the *Converted the ACPI internal object disassemble and display code within the* AML debugger to fully table-driven operation, reducing code size and increasing maintainability.

Fixed a regression with the ConcatenateResTemplate() ASL operator introduced *Fixed a regression with the ConcatenateResTemplate() ASL operator introduced* in the 20051021 release.

Implemented support for "local" internal ACPI object types within the debugger "Object" command and the AcpiWalkNamespace external interfaces. **These local types include RegionFields, BankFields, IndexFields, Alias, and** *These local types include RegionFields, BankFields, IndexFields, Alias, and* reference objects.

Moved common AML resource handling code into a new file, "utresrc.c". This *Moved common AML resource handling code into a new file, "utresrc.c". This* code is shared by both the Resource Manager and the AML Debugger.

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the** *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the* code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency** *and data size. Note that these values will vary depending on the efficiency* of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  76.1K Code, 12.2K Data,  88.3K Total
    Debug Version:     163.5K Code, 67.0K Data, 230.5K Total
  Current Release:
    Non-Debug Version:  76.2K Code, 12.3K Data,  88.5K Total
    Debug Version:     163.0K Code, 67.4K Data, 230.4K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem with very large initializer lists (more than 4000 elements) *Fixed a problem with very large initializer lists (more than 4000 elements)* for both Buffer and Package objects where the parse stack could overflow.

Enhanced the pre-compile source code scan for non-ASCII characters to ignore characters within comment fields. The scan is now always performed and is no *Enhanced the pre-compile source code scan for non-ASCII characters to ignore characters within comment fields. The scan is now always performed and is no* longer optional, detecting invalid characters within a source file immediately rather than during the parse phase or later.

Enhanced the ASL grammar definition to force early reductions on all list- *Enhanced the ASL grammar definition to force early reductions on all list-*

style grammar elements so that the overall parse stack usage is greatly reduced. This should improve performance and reduce the possibility of parse
reduced. This should improve performance and reduce the possibility of parse
stack overflow.

Eliminated all reduce/reduce conflicts in the iASL parser generation. Also,
Eliminated all reduce/reduce conflicts in the iASL parser generation. Also,
with the addition of a %expected statement, the compiler generates from source with no warnings.

Fixed a possible segment fault in the disassembler if the input filename does not contain a "dot" extension (Thomas Renninger).

---
21 October 2005. Summary of changes for version 20051021:

1) ACPI CA Core Subsystem:

Implemented support for the EM64T and other x86-64 processors. This essentially entails recognizing that these processors support non-aligned memory transfers. Previously, all 64-bit processors were assumed to lack hardware support for non-aligned transfers.

Completed conversion of the Resource Manager to nearly full table-driven operation. Specifically, the resource conversion code (convert AML to internal format and the reverse) and the debug code to dump internal **resource descriptors are fully table-driven, reducing code and data size and**
resource descriptors are fully table-driven, reducing code and data size and
improving maintainability.

The OSL interfaces for Acquire and Release Lock now use a 64-bit flag word on 64-bit processors instead of a fixed 32-bit word. (With assistance from
The OSL interfaces for Acquire and Release Lock now use a 64-bit flag word on 64-bit processors instead of a fixed 32-bit word. (With assistance from
Alexey Starikovskiy)

Implemented support within the resource conversion code for the Type-Specific byte within the various ACPI 3.0 *WordSpace macros.

Fixed some issues within the resource conversion code for the type-specific
Fixed some issues within the resource conversion code for the type-specific
flags for both Memory and I/O address resource descriptors. For Memory, implemented support for the MTP and TTP flags. For I/O, split the TRS and TTP flags into two separate fields.

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the**
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the
code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency**
and data size. Note that these values will vary depending on the efficiency
of the compiler and the compiler options used during generation.

  Previous Release:

    Non-Debug Version: 77.1K Code, 12.1K Data, 89.2K Total
    Debug Version: 168.0K Code, 68.3K Data, 236.3K Total
  Current Release:
    Non-Debug Version: 76.1K Code, 12.2K Data, 88.3K Total
    Debug Version: 163.5K Code, 67.0K Data, 230.5K Total

2) iASL Compiler/Disassembler:

Relaxed a compiler restriction that disallowed a ResourceIndex byte if the
Relaxed a compiler restriction that disallowed a ResourceIndex byte if the
corresponding ResourceSource string was not also present in a resource descriptor declaration. This restriction caused problems with existing **AML/ASL code that includes the Index byte without the string. When such AML**
AML/ASL code that includes the Index byte without the string. When such AML
was disassembled, it could not be compiled without modification. Further, **the modified code created a resource template with a different size than the original, breaking code that used fixed offsets into the resource template**
the modified code created a resource template with a different size than the original, breaking code that used fixed offsets into the resource template
buffer.

Removed a recent feature of the disassembler to ignore a lone ResourceIndex
Removed a recent feature of the disassembler to ignore a lone ResourceIndex
byte. This byte is now emitted if present so that the exact AML can be reproduced when the disassembled code is recompiled.

Improved comments and text alignment for the resource descriptor code emitted by the disassembler.

Implemented disassembler support for the ACPI 3.0 AccessSize field within a
Implemented disassembler support for the ACPI 3.0 AccessSize field within a
Register() resource descriptor.

---
30 September 2005. Summary of changes for version 20050930:

1) ACPI CA Core Subsystem:

Completed a major overhaul of the Resource Manager code - specifically, **optimizations in the area of the AML/internal resource conversion code. The code has been optimized to simplify and eliminate duplicated code, CPU stack**
optimizations in the area of the AML/internal resource conversion code. The code has been optimized to simplify and eliminate duplicated code, CPU stack
use has been decreased by optimizing function parameters and local **variables, and naming conventions across the manager have been standardized**
variables, and naming conventions across the manager have been standardized
for clarity and ease of maintenance (this includes function, parameter, variable, and struct/typedef names.) The update may force changes in some driver code, depending on how resources are handled by the host OS.

All Resource Manager dispatch and information tables have been moved to a single location for clarity and ease of maintenance. One new file was created, named "rsinfo.c".

The ACPI return macros (return_ACPI_STATUS, etc.) have been modified to

guarantee that the argument is not evaluated twice, making them less prone
guarantee that the argument is not evaluated twice, making them less prone to macro side-effects. However, since there exists the possibility of **additional stack use if a particular compiler cannot optimize them (such as in the debug generation case), the original macros are optionally available.**
additional stack use if a particular compiler cannot optimize them (such as in the debug generation case), the original macros are optionally available. Note that some invocations of the return_VALUE macro may now cause size **mismatch warnings; the return_UINT8 and return_UINT32 macros are provided to**
mismatch warnings; the return_UINT8 and return_UINT32 macros are provided to eliminate these. (From Randy Dunlap)

Implemented a new mechanism to enable debug tracing for individual control
Implemented a new mechanism to enable debug tracing for individual control methods. A new external interface, AcpiDebugTrace, is provided to enable this mechanism. The intent is to allow the host OS to easily enable and disable tracing for problematic control methods. This interface can be easily exposed to a user or debugger interface if desired. See the file psxface.c for details.

AcpiUtCallocate will now return a valid pointer if a length of zero is specified - a length of one is used and a warning is issued. This matches the behavior of AcpiUtAllocate.

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the**
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency**
and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.5K Code, 12.0K Data,  89.5K Total
    Debug Version:     168.1K Code, 68.4K Data, 236.5K Total
  Current Release:
    Non-Debug Version:  77.1K Code, 12.1K Data,  89.2K Total
    Debug Version:     168.0K Code, 68.3K Data, 236.3K Total
```

2) iASL Compiler/Disassembler:

A remark is issued if the effective compile-time length of a package or buffer is zero. Previously, this was a warning.

---

16 September 2005. Summary of changes for version 20050916:

1) ACPI CA Core Subsystem:

Fixed a problem within the Resource Manager where support for the Generic **Register descriptor was not fully implemented. This descriptor is now fully**
Register descriptor was not fully implemented. This descriptor is now fully recognized, parsed, disassembled, and displayed.

Completely restructured the Resource Manager code to utilize table-driven **dispatch and lookup, eliminating many of the large switch() statements. This**
dispatch and lookup, eliminating many of the large switch() statements. This reduces overall subsystem code size and code complexity. Affects the resource parsing and construction, disassembly, and debug dump output.

Cleaned up and restructured the debug dump output for all resource descriptors. Improved readability of the output and reduced code size.

Fixed a problem where changes to internal data structures caused the optional ACPI_MUTEX_DEBUG code to fail compilation if specified.

Code and Data Size: The current and previous library sizes for the core subsystem are shown below. These are the code and data sizes for the **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the**
acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code **and data size. Note that these values will vary depending on the efficiency**
and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.4K Code, 11.8K Data,  90.2K Total
    Debug Version:     169.6K Code, 69.9K Data, 239.5K Total
  Current Release:
    Non-Debug Version:  77.5K Code, 12.0K Data,  89.5K Total
    Debug Version:     168.1K Code, 68.4K Data, 236.5K Total
```

2) iASL Compiler/Disassembler:

Updated the disassembler to automatically insert an EndDependentFn() macro
Updated the disassembler to automatically insert an EndDependentFn() macro into the ASL stream if this macro is missing in the original AML code, simplifying compilation of the resulting ASL module.

Fixed a problem in the disassembler where a disassembled ResourceSource **string (within a large resource descriptor) was not surrounded by quotes and**
string (within a large resource descriptor) was not surrounded by quotes and not followed by a comma, causing errors when the resulting ASL module was compiled. Also, escape sequences within a ResourceSource string are now handled correctly (especially "\\")

---

02 September 2005. Summary of changes for version 20050902:

1) ACPI CA Core Subsystem:

Fixed a problem with the internal Owner ID allocation and deallocation mechanisms for control method execution and recursive method invocation. This should eliminate the OWNER_ID_LIMIT exceptions and "Invalid OwnerId" messages seen on some systems. Recursive method invocation depth is currently limited to 255. (Alexey Starikovskiy)

Completely eliminated all vestiges of support for the "module-level **executable code" until this support is fully implemented and debugged. This**

executable code" until this support is fully implemented and debugged. This should eliminate the NO_RETURN_VALUE exceptions seen during table load on some systems that invoke this support.

Fixed a problem within the resource manager code where the transaction flags
Fixed a problem within the resource manager code where the transaction flags for a 64-bit address descriptor were handled incorrectly in the type-specific flag byte.

Consolidated duplicate code within the address descriptor resource manager
Consolidated duplicate code within the address descriptor resource manager code, reducing overall subsystem code size.

Fixed a fault when using the AML debugger "disassemble" command to disassemble individual control methods.

Removed references to the "release_current" directory within the Unix release package.

Code and Data Size: The current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler. These values do not **include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler** and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.6K Code, 11.7K Data,  90.3K Total
    Debug Version:     170.0K Code, 69.9K Data, 239.9K Total
  Current Release:
    Non-Debug Version:  78.4K Code, 11.8K Data,  90.2K Total
    Debug Version:     169.6K Code, 69.9K Data, 239.5K Total
```

2) iASL Compiler/Disassembler:

Implemented an error check for illegal duplicate values in the interrupt and
Implemented an error check for illegal duplicate values in the interrupt and dma lists for the following ASL macros: Dma(), Irq(), IrqNoFlags(), and Interrupt().

Implemented error checking for the Irq() and IrqNoFlags() macros to detect
Implemented error checking for the Irq() and IrqNoFlags() macros to detect too many values in the interrupt list (16 max) and invalid values in the list (range 0 - 15)

The maximum length string literal within an ASL file is now restricted to 200 characters as per the ACPI specification.

Fixed a fault when using the -ln option (generate namespace listing).

Implemented an error check to determine if a DescriptorName within a resource descriptor has already been used within the current scope.

---

15 August 2005. Summary of changes for version 20050815:

1) ACPI CA Core Subsystem:

Implemented a full bytewise compare to determine if a table load request is attempting to load a duplicate table. The compare is performed if the table
Implemented a full bytewise compare to determine if a table load request is attempting to load a duplicate table. The compare is performed if the table signatures and table lengths match. This will allow different tables with **the same OEM Table ID and revision to be loaded - probably against the ACPI**
the same OEM Table ID and revision to be loaded - probably against the ACPI specification, but discovered in the field nonetheless.

Added the changes.txt logfile to each of the zipped release packages.

Code and Data Size: Current and previous core subsystem library sizes are **shown below. These are the code and data sizes for the acpica.lib produced**
shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the **debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and**
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.6K Code, 11.7K Data,  90.3K Total
    Debug Version:     167.0K Code, 69.9K Data, 236.9K Total
  Current Release:
    Non-Debug Version:  78.6K Code, 11.7K Data,  90.3K Total
    Debug Version:     170.0K Code, 69.9K Data, 239.9K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem where incorrect AML code could be generated for Package objects if optimization is disabled (via the -oa switch).

Fixed a problem with where incorrect AML code is generated for variable-**length packages when the package length is not specified and the number of**
length packages when the package length is not specified and the number of initializer values is greater than 255.

---

29 July 2005. Summary of changes for version 20050729:

1) ACPI CA Core Subsystem:

Implemented support to ignore an attempt to install/load a particular ACPI
Implemented support to ignore an attempt to install/load a particular ACPI table more than once. Apparently there exists BIOS code that repeatedly attempts to load the same SSDT upon certain events. With assistance from Venkatesh Pallipadi.

Restructured the main interface to the AML parser in order to correctly **handle all exceptional conditions. This will prevent leakage of the**

OwnerId
resource and should eliminate the AE_OWNER_ID_LIMIT exceptions seen on
some
handle all exceptional conditions. This will prevent leakage of the OwnerId
resource and should eliminate the AE_OWNER_ID_LIMIT exceptions seen on some
machines. With assistance from Alexey Starikovskiy.

Support for "module level code" has been disabled in this version due to
a
number of issues that have appeared on various machines. The support can
be
Support for "module level code" has been disabled in this version due to a
number of issues that have appeared on various machines. The support can be
enabled by defining ACPI_ENABLE_MODULE_LEVEL_CODE during subsystem
compilation. When the issues are fully resolved, the code will be enabled
by
compilation. When the issues are fully resolved, the code will be enabled by
default again.

Modified the internal functions for debug print support to define the
FunctionName parameter as a (const char *) for compatibility with
compiler
*FunctionName parameter as a (const char *) for compatibility with compiler*
built-in macros such as __FUNCTION__, etc.

Linted the entire ACPICA source tree for both 32-bit and 64-bit.

Implemented support to display an object count summary for the AML
Debugger
Implemented support to display an object count summary for the AML Debugger
commands Object and Methods.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.6K Code, 11.6K Data,  90.2K Total
    Debug Version:     170.0K Code, 69.7K Data, 239.7K Total
  Current Release:
    Non-Debug Version:  78.6K Code, 11.7K Data,  90.3K Total
    Debug Version:     167.0K Code, 69.9K Data, 236.9K Total
```

2) iASL Compiler/Disassembler:

Fixed a regression that appeared in the 20050708 version of the compiler
where an error message was inadvertently emitted for invocations of the
_OSI
where an error message was inadvertently emitted for invocations of the _OSI
reserved control method.

---
08 July 2005. Summary of changes for version 20050708:

1) ACPI CA Core Subsystem:

The use of the CPU stack in the debug version of the subsystem has been
considerably reduced. Previously, a debug structure was declared in every
function that used the debug macros. This structure has been removed in
favor of declaring the individual elements as parameters to the debug
functions. This reduces the cumulative stack use during nested execution
of
ACPI function calls at the cost of a small increase in the code size of
the
debug version of the subsystem. With assistance from Alexey Starikovskiy
and
functions. This reduces the cumulative stack use during nested execution of
ACPI function calls at the cost of a small increase in the code size of the
debug version of the subsystem. With assistance from Alexey Starikovskiy and
Len Brown.

Added the ACPI_GET_FUNCTION_NAME macro to enable the compiler-dependent
headers to define a macro that will return the current function name at
runtime (such as __FUNCTION__ or _func_, etc.) The function name is used
by
runtime (such as __FUNCTION__ or _func_, etc.) The function name is used by
the debug trace output. If ACPI_GET_FUNCTION_NAME is not defined in the
compiler-dependent header, the function name is saved on the CPU stack
(one
compiler-dependent header, the function name is saved on the CPU stack (one
pointer per function.) This mechanism is used because apparently there
exists no standard ANSI-C defined macro that that returns the function
name.
exists no standard ANSI-C defined macro that that returns the function name.

Redesigned and reimplemented the "Owner ID" mechanism used to track
namespace objects created/deleted by ACPI tables and control method
execution. A bitmap is now used to allocate and free the IDs, thus
solving
the wraparound problem present in the previous implementation. The size
of
execution. A bitmap is now used to allocate and free the IDs, thus solving
the wraparound problem present in the previous implementation. The size of
the namespace node descriptor was reduced by 2 bytes as a result (Alexey
Starikovskiy).

Removed the UINT32_BIT and UINT16_BIT types that were used for the
bitfield
Removed the UINT32_BIT and UINT16_BIT types that were used for the bitfield
flag definitions within the headers for the predefined ACPI tables. These
have been replaced by UINT8_BIT in order to increase the code portability
of
have been replaced by UINT8_BIT in order to increase the code portability of
the subsystem. If the use of UINT8 remains a problem, we may be forced to
eliminate bitfields entirely because of a lack of portability.

Enhanced the performance of the AcpiUtUpdateObjectReference procedure.
This
is a frequently used function and this improvement increases the
performance
Enhanced the performance of the AcpiUtUpdateObjectReference procedure. This
is a frequently used function and this improvement increases the performance
of the entire subsystem (Alexey Starikovskiy).

Fixed several possible memory leaks and the inverse - premature object
deletion (Alexey Starikovskiy).

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced

by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.6K Code, 11.5K Data,  90.1K Total
    Debug Version:     165.2K Code, 69.6K Data, 234.8K Total
  Current Release:
    Non-Debug Version:  78.6K Code, 11.6K Data,  90.2K Total
    Debug Version:     170.0K Code, 69.7K Data, 239.7K Total
```

---

24 June 2005. Summary of changes for version 20050624:

1) ACPI CA Core Subsystem:

Modified the new OSL cache interfaces to use ACPI_CACHE_T as the type for
the host-defined cache object. This allows the OSL implementation to
define
the host-defined cache object. This allows the OSL implementation to define
and type this object in any manner desired, simplifying the OSL
implementation. For example, ACPI_CACHE_T is defined as kmem_cache_t for
Linux, and should be defined in the OS-specific header file for other
operating systems as required.

Changed the interface to AcpiOsAcquireObject to directly return the
requested object as the function return (instead of ACPI_STATUS.) This
change was made for performance reasons, since this is the purpose of the
interface in the first place. AcpiOsAcquireObject is now similar to the
AcpiOsAllocate interface.

Implemented a new AML debugger command named Businfo. This command
displays
information about all devices that have an associate _PRT object. The
_ADR,
Implemented a new AML debugger command named Businfo. This command displays
information about all devices that have an associate _PRT object. The _ADR,
_HID, _UID, and _CID are displayed for these devices.

Modified the initialization sequence in AcpiInitializeSubsystem to call
the
OSL interface AcpiOslInitialize first, before any local initialization.
This
Modified the initialization sequence in AcpiInitializeSubsystem to call the
OSL interface AcpiOslInitialize first, before any local initialization. This
change was required because the global initialization now calls OSL
interfaces.

Enhanced the Dump command to display the entire contents of Package
objects
Enhanced the Dump command to display the entire contents of Package objects
(including all sub-objects and their values.)

Restructured the code base to split some files because of size and/or
because the code logically belonged in a separate file. New files are
listed
below. All makefiles and project files included in the ACPI CA release
have
because the code logically belonged in a separate file. New files are listed
below. All makefiles and project files included in the ACPI CA release have

been updated.

```
    utilities/utcache.c           /* Local cache interfaces */
    utilities/utmutex.c           /* Local mutex support */
    utilities/utstate.c           /* State object support */
    interpreter/parser/psloop.c   /* Main AML parse loop */
```

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.3K Code, 11.6K Data,  89.9K Total
    Debug Version:     164.0K Code, 69.1K Data, 233.1K Total
  Current Release:
    Non-Debug Version:  78.6K Code, 11.5K Data,  90.1K Total
    Debug Version:     165.2K Code, 69.6K Data, 234.8K Total
```

2) iASL Compiler/Disassembler:

Fixed a regression introduced in version 20050513 where the use of a
Package
Fixed a regression introduced in version 20050513 where the use of a Package
object within a Case() statement caused a compile time exception. The
original behavior has been restored (a Match() operator is emitted.)

---

17 June 2005. Summary of changes for version 20050617:

1) ACPI CA Core Subsystem:

Moved the object cache operations into the OS interface layer (OSL) to
allow
Moved the object cache operations into the OS interface layer (OSL) to allow
the host OS to handle these operations if desired (for example, the Linux
OSL will invoke the slab allocator). This support is optional; the
compile
time define ACPI_USE_LOCAL_CACHE may be used to utilize the original
cache
OSL will invoke the slab allocator). This support is optional; the compile
time define ACPI_USE_LOCAL_CACHE may be used to utilize the original cache
code in the ACPI CA core. The new OSL interfaces are shown below. See
utalloc.c for an example implementation, and acpiosxf.h for the exact
interface definitions. With assistance from Alexey Starikovskiy.

```
    AcpiOsCreateCache
    AcpiOsDeleteCache
    AcpiOsPurgeCache
    AcpiOsAcquireObject
    AcpiOsReleaseObject
```

Modified the interfaces to AcpiOsAcquireLock and AcpiOsReleaseLock to
return
Modified the interfaces to AcpiOsAcquireLock and AcpiOsReleaseLock to return
and restore a flags parameter. This fits better with many OS lock models.
Note: the current execution state (interrupt handler or not) is no longer
passed to these interfaces. If necessary, the OSL must determine this

state
passed to these interfaces. If necessary, the OSL must determine this state
by itself, a simple and fast operation. With assistance from Alexey
Starikovskiy.

Fixed a problem in the ACPI table handling where a valid XSDT was assumed
present if the revision of the RSDP was 2 or greater. According to the
ACPI
present if the revision of the RSDP was 2 or greater. According to the ACPI
specification, the XSDT is optional in all cases, and the table manager
therefore now checks for both an RSDP >=2 and a valid XSDT pointer.
Otherwise, the RSDT pointer is used. Some ACPI 2.0 compliant BIOSs
contain
Otherwise, the RSDT pointer is used. Some ACPI 2.0 compliant BIOSs contain
only the RSDT.

Fixed an interpreter problem with the Mid() operator in the case of an
input
string where the resulting output string is of zero length. It now
correctly
Fixed an interpreter problem with the Mid() operator in the case of an input
string where the resulting output string is of zero length. It now correctly
returns a valid, null terminated string object instead of a string object
with a null pointer.

Fixed a problem with the control method argument handling to allow a
store
to an Arg object that already contains an object of type Device. The
Device
Fixed a problem with the control method argument handling to allow a store
to an Arg object that already contains an object of type Device. The Device
object is now correctly overwritten. Previously, an error was returned.

Enhanced the debugger Find command to emit object values in addition to
the
found object pathnames. The output format is the same as the dump
namespace
Enhanced the debugger Find command to emit object values in addition to the
found object pathnames. The output format is the same as the dump namespace
command.

Enhanced the debugger Set command. It now has the ability to set the
value
of any Named integer object in the namespace (Previously, only method
locals
Enhanced the debugger Set command. It now has the ability to set the value
of any Named integer object in the namespace (Previously, only method locals
and args could be set.)

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

  Previous Release:
    Non-Debug Version: 78.1K Code, 11.6K Data, 89.7K Total

    Debug Version: 164.0K Code, 69.3K Data, 233.3K Total
  Current Release:
    Non-Debug Version: 78.3K Code, 11.6K Data, 89.9K Total
    Debug Version: 164.0K Code, 69.1K Data, 233.1K Total

2) iASL Compiler/Disassembler:

Fixed a regression in the disassembler where if/else/while constructs
were
Fixed a regression in the disassembler where if/else/while constructs were
output incorrectly. This problem was introduced in the previous release
(20050526). This problem also affected the single-step disassembly in the
debugger.

Fixed a problem where compiling the reserved _OSI method would randomly
(but
Fixed a problem where compiling the reserved _OSI method would randomly (but
rarely) produce compile errors.

Enhanced the disassembler to emit compilable code in the face of
incorrect
Enhanced the disassembler to emit compilable code in the face of incorrect
AML resource descriptors. If the optional ResourceSourceIndex is present,
but the ResourceSource is not, do not emit the ResourceSourceIndex in the
disassembly. Otherwise, the resulting code cannot be compiled without
errors.

---
26 May 2005. Summary of changes for version 20050526:

1) ACPI CA Core Subsystem:

Implemented support to execute Type 1 and Type 2 AML opcodes appearing at
the module level (not within a control method.) These opcodes are
executed
exactly once at the time the table is loaded. This type of code was legal
up
until the release of ACPI 2.0B (2002) and is now supported within ACPI CA
in
order to provide backwards compatibility with earlier BIOS
implementations.
the module level (not within a control method.) These opcodes are executed
exactly once at the time the table is loaded. This type of code was legal up
until the release of ACPI 2.0B (2002) and is now supported within ACPI CA in
order to provide backwards compatibility with earlier BIOS implementations.
This eliminates the "Encountered executable code at module level" warning
that was previously generated upon detection of such code.

Fixed a problem in the interpreter where an AE_NOT_FOUND exception could
inadvertently be generated during the lookup of namespace objects in the
second pass parse of ACPI tables and control methods. It appears that
this
problem could occur during the resolution of forward references to
namespace
second pass parse of ACPI tables and control methods. It appears that this
problem could occur during the resolution of forward references to namespace
objects.

Added the ACPI_MUTEX_DEBUG #ifdef to the AcpiUtReleaseMutex function,
corresponding to the same #ifdef in the AcpiUtAcquireMutex function. This
allows the deadlock detection debug code to be compiled out in the normal
case, improving mutex performance (and overall subsystem performance)
considerably.

Implemented a handful of miscellaneous fixes for possible memory leaks on

error conditions and error handling control paths. These fixes were
suggested by FreeBSD and the Coverity Prevent source code analysis tool.

Added a check for a null RSDT pointer in AcpiGetFirmwareTable
(tbxfroot.c)
Added a check for a null RSDT pointer in AcpiGetFirmwareTable (tbxfroot.c)
to prevent a fault in this error case.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.2K Code, 11.6K Data,  89.8K Total
    Debug Version:     163.7K Code, 69.3K Data, 233.0K Total
  Current Release:
    Non-Debug Version:  78.1K Code, 11.6K Data,  89.7K Total
    Debug Version:     164.0K Code, 69.3K Data, 233.3K Total
```

2) iASL Compiler/Disassembler:

Implemented support to allow Type 1 and Type 2 ASL operators to appear at
the module level (not within a control method.) These operators will be
executed once at the time the table is loaded. This type of code was
legal
executed once at the time the table is loaded. This type of code was legal
up until the release of ACPI 2.0B (2002) and is now supported by the iASL
compiler in order to provide backwards compatibility with earlier BIOS
ASL
compiler in order to provide backwards compatibility with earlier BIOS ASL
code.

The ACPI integer width (specified via the table revision ID or the -r
override, 32 or 64 bits) is now used internally during compile-time
constant
override, 32 or 64 bits) is now used internally during compile-time constant
folding to ensure that constants are truncated to 32 bits if necessary.
Previously, the revision ID value was only emitted in the AML table
header.
Previously, the revision ID value was only emitted in the AML table header.

An error message is now generated for the Mutex and Method operators if
the
An error message is now generated for the Mutex and Method operators if the
SyncLevel parameter is outside the legal range of 0 through 15.

Fixed a problem with the Method operator ParameterTypes list handling
(ACPI
3.0). Previously, more than 2 types or 2 arguments generated a syntax
error.
Fixed a problem with the Method operator ParameterTypes list handling (ACPI
3.0). Previously, more than 2 types or 2 arguments generated a syntax error.
The actual underlying implementation of method argument typechecking is
still under development, however.


```
---------------------------------------
```
13 May 2005. Summary of changes for version 20050513:

1) ACPI CA Core Subsystem:

Implemented support for PCI Express root bridges -- added support for
device
Implemented support for PCI Express root bridges -- added support for device
PNP0A08 in the root bridge search within AcpiEvPciConfigRegionSetup.

The interpreter now automatically truncates incoming 64-bit constants to
32
bits if currently executing out of a 32-bit ACPI table (Revision < 2).
This
The interpreter now automatically truncates incoming 64-bit constants to 32
bits if currently executing out of a 32-bit ACPI table (Revision < 2). This
also affects the iASL compiler constant folding. (Note: as per below, the
iASL compiler no longer allows 64-bit constants within 32-bit tables.)

Fixed a problem where string and buffer objects with "static" pointers
(pointers to initialization data within an ACPI table) were not handled
consistently. The internal object copy operation now always copies the
data
consistently. The internal object copy operation now always copies the data
to a newly allocated buffer, regardless of whether the source object is
static or not.

Fixed a problem with the FromBCD operator where an implicit result
conversion was improperly performed while storing the result to the
target
conversion was improperly performed while storing the result to the target
operand. Since this is an "explicit conversion" operator, the implicit
conversion should never be performed on the output.

Fixed a problem with the CopyObject operator where a copy to an existing
named object did not always completely overwrite the existing object
stored
at name. Specifically, a buffer-to-buffer copy did not delete the
existing
named object did not always completely overwrite the existing object stored
at name. Specifically, a buffer-to-buffer copy did not delete the existing
buffer.

Replaced "InterruptLevel" with "InterruptNumber" in all GPE interfaces
and
Replaced "InterruptLevel" with "InterruptNumber" in all GPE interfaces and
structs for consistency.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.2K Code, 11.6K Data,  89.8K Total
    Debug Version:     163.7K Code, 69.3K Data, 233.0K Total
  Current Release: (Same sizes)
```

```
8711      Non-Debug Version:  78.2K Code, 11.6K Data,  89.8K Total
8712      Debug Version:     163.7K Code, 69.3K Data, 233.0K Total


8715 2) iASL Compiler/Disassembler:

8717 The compiler now emits a warning if an attempt is made to generate a 64-
8718 bit
8719 integer constant from within a 32-bit ACPI table (Revision < 2). The
8720 integer
5024 The compiler now emits a warning if an attempt is made to generate a 64-bit
5025 integer constant from within a 32-bit ACPI table (Revision < 2). The integer
8721 is truncated to 32 bits.

8723 Fixed a problem with large package objects: if the static length of the
8724 package is greater than 255, the "variable length package" opcode is
8725 emitted. Previously, this caused an error. This requires an update to the
8726 ACPI spec, since it currently (incorrectly) states that packages larger
8727 than
5031 ACPI spec, since it currently (incorrectly) states that packages larger than
8728 255 elements are not allowed.

8730 The disassembler now correctly handles variable length packages and
8731 packages
5034 The disassembler now correctly handles variable length packages and packages
8732 larger than 255 elements.

8734 ----------------------------------------
8735 08 April 2005.  Summary of changes for version 20050408:

8737 1) ACPI CA Core Subsystem:

8739 Fixed three cases in the interpreter where an "index" argument to an ASL
8740 function was still (internally) 32 bits instead of the required 64 bits.
8741 This was the Index argument to the Index, Mid, and Match operators.

8743 The "strupr" function is now permanently local (AcpiUtStrupr), since this
8744 is
5046 The "strupr" function is now permanently local (AcpiUtStrupr), since this is
8745 not a POSIX-defined function and not present in most kernel-level C
8746 libraries. All references to the C library strupr function have been
8747 removed
5048 libraries. All references to the C library strupr function have been removed
8748 from the headers.

8750 Completed the deployment of static functions/prototypes. All prototypes
8751 with
8752 the static attribute have been moved from the headers to the owning C
8753 file.
5051 Completed the deployment of static functions/prototypes. All prototypes with
5052 the static attribute have been moved from the headers to the owning C file.

8755 Implemented an extract option (-e) for the AcpiBin utility (AML binary
8756 utility). This option allows the utility to extract individual ACPI
8757 tables
5055 utility). This option allows the utility to extract individual ACPI tables
8758 from the output of AcpiDmp. It provides the same functionality of the
8759 acpixtract.pl perl script without the worry of setting the correct perl
8760 options. AcpiBin runs on Windows and has not yet been generated/validated
8761 in
5058 options. AcpiBin runs on Windows and has not yet been generated/validated in
8762 the Linux/Unix environment (but should be soon).
8763
8764 Updated and fixed the table dump option for AcpiBin (-d). This option
8765 converts a single ACPI table to a hex/ascii file, similar to the output
8766 of
```



```
5062 converts a single ACPI table to a hex/ascii file, similar to the output of
8767 AcpiDmp.

8769 Code and Data Size: Current and previous core subsystem library sizes are
8770 shown below. These are the code and data sizes for the acpica.lib
8771 produced
5066 shown below. These are the code and data sizes for the acpica.lib produced
8772 by the Microsoft Visual C++ 6.0 compiler, and these values do not include
8773 any ACPI driver or OSPM code. The debug version of the code includes the
8774 debug output trace mechanism and has a much larger code and data size.
8775 Note
8776 that these values will vary depending on the efficiency of the compiler
8777 and
5069 debug output trace mechanism and has a much larger code and data size. Note
5070 that these values will vary depending on the efficiency of the compiler and
8778 the compiler options used during generation.

8780   Previous Release:
8781     Non-Debug Version:  78.0K Code, 11.6K Data,  89.6K Total
8782     Debug Version:     163.5K Code, 69.3K Data, 232.8K Total
8783   Current Release:
8784     Non-Debug Version:  78.2K Code, 11.6K Data,  89.8K Total
8785     Debug Version:     163.7K Code, 69.3K Data, 233.0K Total


8788 2) iASL Compiler/Disassembler:

8790 Disassembler fix: Added a check to ensure that the table length found in
8791 the
8792 ACPI table header within the input file is not longer than the actual
8793 input
5083 Disassembler fix: Added a check to ensure that the table length found in the
5084 ACPI table header within the input file is not longer than the actual input
8794 file size. This indicates some kind of file or table corruption.

8796 ----------------------------------------
8797 29 March 2005.  Summary of changes for version 20050329:

8799 1) ACPI CA Core Subsystem:

8801 An error is now generated if an attempt is made to create a Buffer Field
8802 of
5092 An error is now generated if an attempt is made to create a Buffer Field of
8803 length zero (A CreateField with a length operand of zero.)

8805 The interpreter now issues a warning whenever executable code at the
8806 module
5095 The interpreter now issues a warning whenever executable code at the module
8807 level is detected during ACPI table load. This will give some idea of the
8808 prevalence of this type of code.

8810 Implemented support for references to named objects (other than control
8811 methods) within package objects.

8813 Enhanced package object output for the debug object. Package objects are
8814 now
5102 Enhanced package object output for the debug object. Package objects are now
8815 completely dumped, showing all elements.

8817 Enhanced miscellaneous object output for the debug object. Any object can
8818 now be written to the debug object (for example, a device object can be
8819 written, and the type of the object will be displayed.)

8821 The "static" qualifier has been added to all local functions across both
8822 the
5109 The "static" qualifier has been added to all local functions across both the
```

core subsystem and the iASL compiler.

The number of "long" lines (> 80 chars) within the source has been significantly reduced, by about 1/3.

Cleaned up all header files to ensure that all CA/iASL functions are prototyped (even static functions) and the formatting is consistent.

Two new header files have been added, acopcode.h and acnames.h.

Removed several obsolete functions that were no longer used.

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.3K Code, 11.5K Data,  89.8K Total
    Debug Version:     165.4K Code, 69.7K Data, 236.1K Total
  Current Release:
    Non-Debug Version:  78.0K Code, 11.6K Data,  89.6K Total
    Debug Version:     163.5K Code, 69.3K Data, 232.8K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem with the resource descriptor generation/support. For the ResourceSourceIndex and the ResourceSource fields, both must be present, or both must be not present - can't have one without the other.

The compiler now returns non-zero from the main procedure if any errors have occurred during the compilation.

---

09 March 2005. Summary of changes for version 20050309:

1) ACPI CA Core Subsystem:

The string-to-buffer implicit conversion code has been modified again after a change to the ACPI specification. In order to match the behavior of the other major ACPI implementation, the target buffer is no longer truncated if the source string is smaller than an existing target buffer. This change requires an update to the ACPI spec, and should eliminate the recent AE_AML_BUFFER_LIMIT issues.

The "implicit return" support was rewritten to a new algorithm that solves the general case. Rather than attempt to determine when a method is about to exit, the result of every ASL operator is saved momentarily until the very next ASL operator is executed. Therefore, no matter how the method exits, there will always be a saved implicit return value. This feature is only enabled with the AcpiGbl_EnableInterpreterSlack flag, and should eliminate AE_AML_NO_RETURN_VALUE errors when enabled.

Implemented implicit conversion support for the predicate (operand) of the If, Else, and While operators. String and Buffer arguments are automatically converted to Integers.

Changed the string-to-integer conversion behavior to match the new ACPI errata: "If no integer object exists, a new integer is created. The ASCII string is interpreted as a hexadecimal constant. Each string character is interpreted as a hexadecimal value ('0'-'9', 'A'-'F', 'a', 'f'), starting with the first character as the most significant digit, and ending with the first non-hexadecimal character or end-of-string." This means that the first non-hex character terminates the conversion and this is the code that was changed.

Fixed a problem where the ObjectType operator would fail (fault) when used on an Index of a Package which pointed to a null package element. The operator now properly returns zero (Uninitialized) in this case.

Fixed a problem where the While operator used excessive memory by not properly popping the result stack during execution. There was no memory leak after execution, however. (Code provided by Valery Podrezov.)

Fixed a problem where references to control methods within Package objects caused the method to be invoked, instead of producing a reference object pointing to the method.

Restructured and simplified the pswalk.c module (AcpiPsDeleteParseTree) to improve performance and reduce code size. (Code provided by Alexey Starikovskiy.)

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced

by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.3K Code, 11.5K Data,  89.8K Total
    Debug Version:     165.4K Code, 69.6K Data, 236.0K Total
  Current Release:
    Non-Debug Version:  78.3K Code, 11.5K Data,  89.8K Total
    Debug Version:     165.4K Code, 69.7K Data, 236.1K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem with the Return operator with no arguments. Since the AML
grammar for the byte encoding requires an operand for the Return opcode,
the
grammar for the byte encoding requires an operand for the Return opcode, the
compiler now emits a Return(Zero) for this case. An ACPI specification
update has been written for this case.

For tables other than the DSDT, namepath optimization is automatically
disabled. This is because SSDTs can be loaded anywhere in the namespace,
the
disabled. This is because SSDTs can be loaded anywhere in the namespace, the
compiler has no knowledge of where, and thus cannot optimize namepaths.

Added "ProcessorObj" to the ObjectTypeKeyword list. This object type was
inadvertently omitted from the ACPI specification, and will require an
update to the spec.

The source file scan for ASCII characters is now optional (-a). This
change
The source file scan for ASCII characters is now optional (-a). This change
was made because some vendors place non-ascii characters within comments.
However, the scan is simply a brute-force byte compare to ensure all
characters in the file are in the range 0x00 to 0x7F.

Fixed a problem with the CondRefOf operator where the compiler was
inappropriately checking for the existence of the target. Since the point
of
inappropriately checking for the existence of the target. Since the point of
the operator is to check for the existence of the target at run-time, the
compiler no longer checks for the target existence.

Fixed a problem where errors generated from the internal AML interpreter
during constant folding were not handled properly, causing a fault.

Fixed a problem with overly aggressive range checking for the Stall
operator. The valid range (max 255) is now only checked if the operand is
of
operator. The valid range (max 255) is now only checked if the operand is of
type Integer. All other operand types cannot be statically checked.

Fixed a problem where control method references within the RefOf,
DeRefOf,
and ObjectType operators were not treated properly. They are now treated
as
Fixed a problem where control method references within the RefOf, DeRefOf,
and ObjectType operators were not treated properly. They are now treated as

actual references, not method invocations.

Fixed and enhanced the "list namespace" option (-ln). This option was
broken
Fixed and enhanced the "list namespace" option (-ln). This option was broken
a number of releases ago.

Improved error handling for the Field, IndexField, and BankField
operators.
Improved error handling for the Field, IndexField, and BankField operators.
The compiler now cleanly reports and recovers from errors in the field
component (FieldUnit) list.

Fixed a disassembler problem where the optional ResourceDescriptor fields
TRS and TTP were not always handled correctly.

Disassembler - Comments in output now use "//" instead of "/*"

```
---------------------------------------
```

28 February 2005. Summary of changes for version 20050228:

1) ACPI CA Core Subsystem:

Fixed a problem where the result of an Index() operator (an object
reference) must increment the reference count on the target object for
the
reference) must increment the reference count on the target object for the
life of the object reference.

Implemented AML Interpreter and Debugger support for the new ACPI 3.0
Extended Address (IO, Memory, Space), QwordSpace, DwordSpace, and
WordSpace
Extended Address (IO, Memory, Space), QwordSpace, DwordSpace, and WordSpace
resource descriptors.

Implemented support in the _OSI method for the ACPI 3.0 "Extended Address
Space Descriptor" string, indicating interpreter support for the
descriptors
Space Descriptor" string, indicating interpreter support for the descriptors
above.

Implemented header support for the new ACPI 3.0 FADT flag bits.

Implemented header support for the new ACPI 3.0 PCI Express bits for the
PM1
Implemented header support for the new ACPI 3.0 PCI Express bits for the PM1
status/enable registers.

Updated header support for the MADT processor local Apic struct and MADT
platform interrupt source struct for new ACPI 3.0 fields.

Implemented header support for the SRAT and SLIT ACPI tables.

Implemented the -s switch in AcpiExec to enable the "InterpreterSlack"
flag
Implemented the -s switch in AcpiExec to enable the "InterpreterSlack" flag
at runtime.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note

that these values will vary depending on the efficiency of the compiler and debug output trace mechanism and has a much larger code and data size. Note the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.2K Code, 11.5K Data,  89.7K Total
    Debug Version:     164.9K Code, 69.2K Data, 234.1K Total
  Current Release:
    Non-Debug Version:  78.3K Code, 11.5K Data,  89.8K Total
    Debug Version:     165.4K Code, 69.6K Data, 236.0K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem with the internal 64-bit String-to-integer conversion with strings less than two characters long.

Fixed a problem with constant folding where the result of the Index() operator can not be considered a constant. This means that Index() cannot be a type3 opcode and this will require an update to the ACPI specification.

Disassembler: Implemented support for the TTP, MTP, and TRS resource descriptor fields. These fields were inadvertently ignored and not output in the disassembly of the resource descriptor.

---

11 February 2005. Summary of changes for version 20050211:

1) ACPI CA Core Subsystem:

Implemented ACPI 3.0 support for implicit conversion within the Match() operator. MatchObjects can now be of type integer, buffer, or string instead of just type integer. Package elements are implicitly converted to the type of the MatchObject. This change aligns the behavior of Match() with the behavior of the other logical operators (LLess(), etc.) It also requires an errata change to the ACPI specification as this support was intended for ACPI 3.0, but was inadvertently omitted.

Fixed a problem with the internal implicit "to buffer" conversion. Strings that are converted to buffers will cause buffer truncation if the string is smaller than the target buffer. Integers that are converted to buffers will not cause buffer truncation, only zero extension (both as per the ACPI spec.) The problem was introduced when code was added to truncate the buffer, but this should not be performed in all cases, only the string case.

buffer, but this should not be performed in all cases, only the string case.

Fixed a problem with the Buffer and Package operators where the interpreter would get confused if two such operators were used as operands to an ASL operator (such as LLess(Buffer(1){0},Buffer(1){1}). The internal result stack was not being popped after the execution of these operators, resulting in an AE_NO_RETURN_VALUE exception.

Fixed a problem with constructs of the form Store(Index(...),...). The reference object returned from Index was inadvertently resolved to an actual value. This problem was introduced in version 20050114 when the behavior of Store() was modified to restrict the object types that can be used as the source operand (to match the ACPI specification.)

Reduced excessive stack use within the AcpiGetObjectInfo procedure.

Added a fix to aclinux.h to allow generation of AcpiExec on Linux.

Updated the AcpiSrc utility to add the FADT_DESCRIPTOR_REV2_MINUS struct.

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.1K Code, 11.5K Data,  89.6K Total
    Debug Version:     164.8K Code, 69.2K Data, 234.0K Total
  Current Release:
    Non-Debug Version:  78.2K Code, 11.5K Data,  89.7K Total
    Debug Version:     164.9K Code, 69.2K Data, 234.1K Total
```

2) iASL Compiler/Disassembler:

Fixed a code generation problem in the constant folding optimization code where incorrect code was generated if a constant was reduced to a buffer object (i.e., a reduced type 5 opcode.)

Fixed a typechecking problem for the ToBuffer operator. Caused by an incorrect return type in the internal opcode information table.

---

25 January 2005. Summary of changes for version 20050125:

1) ACPI CA Core Subsystem:

Fixed a recently introduced problem with the Global Lock where the underlying semaphore was not created. This problem was introduced in

version 20050114, and caused an AE_AML_NO_OPERAND exception during an
Acquire() operation on _GL.

The local object cache is now optional, and is disabled by default. Both
AcpiExec and the iASL compiler enable the cache because they run in user
mode and this enhances their performance. #define
ACPI_ENABLE_OBJECT_CACHE
mode and this enhances their performance. #define ACPI_ENABLE_OBJECT_CACHE
to enable the local cache.

Fixed an issue in the internal function AcpiUtEvaluateObject concerning
the
optional "implicit return" support where an error was returned if no
return
object was expected, but one was implicitly returned. AE_OK is now
returned
Fixed an issue in the internal function AcpiUtEvaluateObject concerning the
optional "implicit return" support where an error was returned if no return
object was expected, but one was implicitly returned. AE_OK is now returned
in this case and the implicitly returned object is deleted.
AcpiUtEvaluateObject is only occasionally used, and only to execute
reserved
AcpiUtEvaluateObject is only occasionally used, and only to execute reserved
methods such as _STA and _INI where the return type is known up front.

Fixed a few issues with the internal convert-to-integer code. It now
returns
Fixed a few issues with the internal convert-to-integer code. It now returns
an error if an attempt is made to convert a null string, a string of only
blanks/tabs, or a zero-length buffer. This affects both implicit
conversion
blanks/tabs, or a zero-length buffer. This affects both implicit conversion
and explicit conversion via the ToInteger() operator.

The internal debug code in AcpiUtAcquireMutex has been commented out. It
is
not needed for normal operation and should increase the performance of
the
entire subsystem. The code remains in case it is needed for debug
purposes
The internal debug code in AcpiUtAcquireMutex has been commented out. It is
not needed for normal operation and should increase the performance of the
entire subsystem. The code remains in case it is needed for debug purposes
again.

The AcpiExec source and makefile are included in the Unix/Linux package
for
The AcpiExec source and makefile are included in the Unix/Linux package for
the first time.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.4K Code,  11.5K Data,   89.9K Total
```



```
    Debug Version:     165.4K Code,  69.4K Data,  234.8K Total
  Current Release:
    Non-Debug Version:  78.1K Code,  11.5K Data,   89.6K Total
    Debug Version:     164.8K Code,  69.2K Data,  234.0K Total
```

2) iASL Compiler/Disassembler:

Switch/Case support: A warning is now issued if the type of the Switch
value
Switch/Case support: A warning is now issued if the type of the Switch value
cannot be determined at compile time. For example, Switch(Arg0) will
generate the warning, and the type is assumed to be an integer. As per
the
ACPI spec, use a construct such as Switch(ToInteger(Arg0)) to eliminate
the
generate the warning, and the type is assumed to be an integer. As per the
ACPI spec, use a construct such as Switch(ToInteger(Arg0)) to eliminate the
warning.

Switch/Case support: Implemented support for buffer and string objects as
the switch value. This is an ACPI 3.0 feature, now that LEqual supports
buffers and strings.

Switch/Case support: The emitted code for the LEqual() comparisons now
uses
the switch value as the first operand, not the second. The case value is
now
Switch/Case support: The emitted code for the LEqual() comparisons now uses
the switch value as the first operand, not the second. The case value is now
the second operand, and this allows the case value to be implicitly
converted to the type of the switch value, not the other way around.

Switch/Case support: Temporary variables are now emitted immediately
within
the control method, not at the global level. This means that there are
now
36 temps available per-method, not 36 temps per-module as was the case
with
Switch/Case support: Temporary variables are now emitted immediately within
the control method, not at the global level. This means that there are now
36 temps available per-method, not 36 temps per-module as was the case with
the earlier implementation (_T_0 through _T_9 and _T_A through _T_Z.)

```
----------------------------------------
```
14 January 2005. Summary of changes for version 20050114:

Added 2005 copyright to all module headers. This affects every module in
the core subsystem, iASL compiler, and the utilities.

1) ACPI CA Core Subsystem:

Fixed an issue with the String-to-Buffer conversion code where the string
null terminator was not included in the buffer after conversion, but
there
is existing ASL that assumes the string null terminator is included. This
is
null terminator was not included in the buffer after conversion, but there
is existing ASL that assumes the string null terminator is included. This is
the root of the ACPI_AML_BUFFER_LIMIT regression. This problem was
introduced in the previous version when the code was updated to correctly
set the converted buffer size as per the ACPI specification. The ACPI
spec
is ambiguous and will be updated to specify that the null terminator must
be
set the converted buffer size as per the ACPI specification. The ACPI spec
is ambiguous and will be updated to specify that the null terminator must be

included in the converted buffer. This also affects the ToBuffer() ASL operator.

Fixed a problem with the Mid() ASL/AML operator where it did not work correctly on Buffer objects. Newly created sub-buffers were not being marked as initialized.

Fixed a problem in AcpiTbFindTable where incorrect string compares were performed on the OemId and OemTableId table header fields. These fields are not null terminated, so strncmp is now used instead of strcmp.

Implemented a restriction on the Store() ASL/AML operator to align the behavior with the ACPI specification. Previously, any object could be used as the source operand. Now, the only objects that may be used are Integers, Buffers, Strings, Packages, Object References, and DDB Handles. If necessary, the original behavior can be restored by enabling the EnableInterpreterSlack flag.

Enhanced the optional "implicit return" support to allow an implicit return value from methods that are invoked externally via the AcpiEvaluateObject interface. This enables implicit returns from the _STA and _INI methods, for example.

Changed the Revision() ASL/AML operator to return the current version of the AML interpreter, in the YYYYMMDD format. Previously, it incorrectly returned the supported ACPI version (This is the function of the _REV method).

Updated the _REV predefined method to return the currently supported version of ACPI, now 3.

Implemented batch mode option for the AcpiExec utility (-b).

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.3K Code,  11.5K Data,   89.8K Total
    Debug Version:     165.3K Code,  69.4K Data,  234.7K Total
```



```
  Current Release:
    Non-Debug Version:  78.4K Code,  11.5K Data,   89.9K Total
    Debug Version:     165.4K Code,  69.4K Data,  234.8K Total
```

---

10 December 2004. Summary of changes for version 20041210:

ACPI 3.0 support is nearing completion in both the iASL compiler and the ACPI CA core subsystem.

1) ACPI CA Core Subsystem:

Fixed a problem in the ToDecimalString operator where the resulting string length was incorrectly calculated. The length is now calculated exactly, eliminating incorrect AE_STRING_LIMIT exceptions.

Fixed a problem in the ToHexString operator to allow a maximum 200 character string to be produced.

Fixed a problem in the internal string-to-buffer and buffer-to-buffer copy routine where the length of the resulting buffer was not truncated to the new size (if the target buffer already existed).

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.3K Code,  11.5K Data,   89.8K Total
    Debug Version:     164.7K Code,  68.5K Data,  233.2K Total
  Current Release:
    Non-Debug Version:  78.3K Code,  11.5K Data,   89.8K Total
    Debug Version:     165.3K Code,  69.4K Data,  234.7K Total
```

2) iASL Compiler/Disassembler:

Implemented the new ACPI 3.0 resource template macros - DWordSpace, ExtendedIO, ExtendedMemory, ExtendedSpace, QWordSpace, and WordSpace. Includes support in the disassembler.

Implemented support for the new (ACPI 3.0) parameter to the Register macro, AccessSize.

Fixed a problem where the _HE resource name for the Interrupt macro was referencing bit 0 instead of bit 1.

Implemented check for maximum 255 interrupts in the Interrupt macro.

Fixed a problem with the predefined resource descriptor names where
incorrect AML code was generated if the offset within the resource buffer
was 0 or 1. The optimizer shortened the AML code to a single byte opcode
but did not update the surrounding package lengths.

**Changes to the Dma macro: All channels within the channel list must be
in**
Changes to the Dma macro: All channels within the channel list must be in
the range 0-7. Maximum 8 channels can be specified. BusMaster operand is
optional (default is BusMaster).

Implemented check for maximum 7 data bytes for the VendorShort macro.

**The ReadWrite parameter is now optional for the Memory32 and similar
macros.**
The ReadWrite parameter is now optional for the Memory32 and similar macros.

---
03 December 2004. Summary of changes for version 20041203:

1) ACPI CA Core Subsystem:

**The low-level field insertion/extraction code (exfldio) has been
completely**
The low-level field insertion/extraction code (exfldio) has been completely
rewritten to eliminate unnecessary complexity, bugs, and boundary
conditions.

**Fixed a problem in the ToInteger, ToBuffer, ToHexString, and
ToDecimalString**
Fixed a problem in the ToInteger, ToBuffer, ToHexString, and ToDecimalString
operators where the input operand could be inadvertently deleted if no
conversion was necessary (e.g., if the input to ToInteger was an Integer
object.)

**Fixed a problem with the ToDecimalString and ToHexString where an
incorrect**
Fixed a problem with the ToDecimalString and ToHexString where an incorrect
exception code was returned if the resulting string would be > 200 chars.
AE_STRING_LIMIT is now returned.

Fixed a problem with the Concatenate operator where AE_OK was always
returned, even if the operation failed.

Fixed a problem in oswinxf (used by AcpiExec and iASL) to allow > 128
semaphores to be allocated.

Code and Data Size: Current and previous core subsystem library sizes are
**shown below. These are the code and data sizes for the acpica.lib
produced**
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
**debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and**
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

  Previous Release:
    Non-Debug Version: 78.5K Code, 11.5K Data,   90.0K Total
    Debug Version: 165.2K Code, 68.6K Data, 233.8K Total
  Current Release:

    Non-Debug Version: 78.3K Code, 11.5K Data,   89.8K Total
    Debug Version: 164.7K Code, 68.5K Data, 233.2K Total

2) iASL Compiler/Disassembler:

Fixed typechecking for the ObjectType and SizeOf operators. Problem was
recently introduced in 20041119.

**Fixed a problem with the ToUUID macro where the upper nybble of each
buffer**
Fixed a problem with the ToUUID macro where the upper nybble of each buffer
byte was inadvertently set to zero.

---
19 November 2004. Summary of changes for version 20041119:

1) ACPI CA Core Subsystem:

**Fixed a problem in the internal ConvertToInteger routine where new
integers
were not truncated to 32 bits for 32-bit ACPI tables. This routine
converts**
Fixed a problem in the internal ConvertToInteger routine where new integers
were not truncated to 32 bits for 32-bit ACPI tables. This routine converts
buffers and strings to integers.

**Implemented support to store a value to an Index() on a String object.
This**
Implemented support to store a value to an Index() on a String object. This
is an ACPI 2.0 feature that had not yet been implemented.

**Implemented new behavior for storing objects to individual package
elements
(via the Index() operator). The previous behavior was to invoke the
implicit**
Implemented new behavior for storing objects to individual package elements
(via the Index() operator). The previous behavior was to invoke the implicit
conversion rules if an object was already present at the index. The new
**behavior is to simply delete any existing object and directly store the
new
object. Although the ACPI specification seems unclear on this subject,
other**
behavior is to simply delete any existing object and directly store the new
object. Although the ACPI specification seems unclear on this subject, other
ACPI implementations behave in this manner. (This is the root of the
AE_BAD_HEX_CONSTANT issue.)

**Modified the RSDP memory scan mechanism to support the extended checksum
for**
Modified the RSDP memory scan mechanism to support the extended checksum for
ACPI 2.0 (and above) RSDPs. Note that the search continues until a valid
RSDP signature is found with a valid checksum.

Code and Data Size: Current and previous core subsystem library sizes are
**shown below. These are the code and data sizes for the acpica.lib
produced**
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
**debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and**
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and

the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.5K Code,  11.5K Data,   90.0K Total
    Debug Version:     165.2K Code,  68.6K Data,  233.8K Total
  Current Release:
    Non-Debug Version:  78.5K Code,  11.5K Data,   90.0K Total
    Debug Version:     165.2K Code,  68.6K Data,  233.8K Total
```

2) iASL Compiler/Disassembler:

Fixed a missing semicolon in the aslcompiler.y file.

---

05 November 2004. Summary of changes for version 20041105:

1) ACPI CA Core Subsystem:

Implemented support for FADT revision 2. This was an interim table (between
Implemented support for FADT revision 2. This was an interim table (between ACPI 1.0 and ACPI 2.0) that adds support for the FADT reset register.

Implemented optional support to allow uninitialized LocalX and ArgX **variables in a control method. The variables are initialized to an Integer**
variables in a control method. The variables are initialized to an Integer object with a value of zero. This support is enabled by setting the AcpiGbl_EnableInterpreterSlack flag to TRUE.

Implemented support for Integer objects for the SizeOf operator. Either 4 or 8 is returned, depending on the current integer size (32-bit or 64-bit,
Implemented support for Integer objects for the SizeOf operator. Either 4 or 8 is returned, depending on the current integer size (32-bit or 64-bit, depending on the parent table revision).

Fixed a problem in the implementation of the SizeOf and ObjectType operators
Fixed a problem in the implementation of the SizeOf and ObjectType operators where the operand was resolved to a value too early, causing incorrect return values for some objects.

Fixed some possible memory leaks during exceptional conditions.

Code and Data Size: Current and previous core subsystem library sizes are **shown below. These are the code and data sizes for the acpica.lib produced**
shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the **debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and**
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.0K Code,  11.5K Data,   89.5K Total
    Debug Version:     164.8K Code,  68.6K Data,  233.4K Total
  Current Release:
    Non-Debug Version:  78.5K Code,  11.5K Data,   90.0K Total
```



```
    Debug Version:     165.2K Code,  68.6K Data,  233.8K Total
```

2) iASL Compiler/Disassembler:

Implemented support for all ACPI 3.0 reserved names and methods.

Implemented all ACPI 3.0 grammar elements in the front-end, including support for semicolons.

Implemented the ACPI 3.0 Function() and ToUUID() macros

Fixed a problem in the disassembler where a Scope() operator would not be emitted properly if the target of the scope was in another table.

---

15 October 2004. Summary of changes for version 20041015:

Note: ACPI CA is currently undergoing an in-depth and complete formal evaluation to test/verify the following areas. Other suggestions are welcome. This will result in an increase in the frequency of releases and the number of bug fixes in the next few months.
- Functional tests for all ASL/AML operators
- All implicit/explicit type conversions
- Bit fields and operation regions
- 64-bit math support and 32-bit-only "truncated" math support
- Exceptional conditions, both compiler and interpreter
- Dynamic object deletion and memory leaks
- ACPI 3.0 support when implemented
- External interfaces to the ACPI subsystem

1) ACPI CA Core Subsystem:

Fixed two alignment issues on 64-bit platforms - within debug statements in AcpiEvGpeDetect and AcpiEvCreateGpeBlock. Removed references to the Address
Fixed two alignment issues on 64-bit platforms - within debug statements in AcpiEvGpeDetect and AcpiEvCreateGpeBlock. Removed references to the Address field within the non-aligned ACPI generic address structure.

Fixed a problem in the Increment and Decrement operators where incorrect operand resolution could result in the inadvertent modification of the original integer when the integer is passed into another method as an argument and the arg is then incremented/decremented.

Fixed a problem in the FromBCD operator where the upper 32-bits of a 64-bit
Fixed a problem in the FromBCD operator where the upper 32-bits of a 64-bit BCD number were truncated during conversion.

Fixed a problem in the ToDecimal operator where the length of the resulting string could be set incorrectly too long if the input operand was a Buffer
Fixed a problem in the ToDecimal operator where the length of the resulting string could be set incorrectly too long if the input operand was a Buffer object.

Fixed a problem in the Logical operators (LLess, etc.) where a NULL byte (0)
Fixed a problem in the Logical operators (LLess, etc.) where a NULL byte (0) within a buffer would prematurely terminate a compare between buffer objects.

Added a check for string overflow (>200 characters as per the ACPI
specification) during the Concatenate operator with two string operands.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.8K Code,  11.5K Data,   89.3K Total
    Debug Version:     164.6K Code,  68.5K Data,  233.1K Total
  Current Release:
    Non-Debug Version:  78.0K Code,  11.5K Data,   89.5K Total
    Debug Version:     164.8K Code,  68.6K Data,  233.4K Total
```

2) iASL Compiler/Disassembler:

Allow the use of the ObjectType operator on uninitialized Locals and Args
(returns 0 as per the ACPI specification).

Fixed a problem where the compiler would fault if there was a syntax
error
Fixed a problem where the compiler would fault if there was a syntax error
in the FieldName of all of the various CreateXXXField operators.

Disallow the use of lower case letters within the EISAID macro, as per
the
ACPI specification. All EISAID strings must be of the form "UUUNNNN"
Where
Disallow the use of lower case letters within the EISAID macro, as per the
ACPI specification. All EISAID strings must be of the form "UUUNNNN" Where
U is an uppercase letter and N is a hex digit.

---
06 October 2004. Summary of changes for version 20041006:

1) ACPI CA Core Subsystem:

Implemented support for the ACPI 3.0 Timer operator. This ASL function
implements a 64-bit timer with 100 nanosecond granularity.

Defined a new OSL interface, AcpiOsGetTimer. This interface is used to
implement the ACPI 3.0 Timer operator. This allows the host OS to
implement
the timer with the best clock available. Also, it keeps the core
subsystem
implement the ACPI 3.0 Timer operator. This allows the host OS to implement
the timer with the best clock available. Also, it keeps the core subsystem
out of the clock handling business, since the host OS (usually) performs
this function.

Fixed an alignment issue on 64-bit platforms. The HwLowLevelRead(Write)
functions use a 64-bit address which is part of the packed ACPI Generic
Address Structure. Since the structure is non-aligned, the alignment

macros
Address Structure. Since the structure is non-aligned, the alignment macros
are now used to extract the address to a local variable before use.

Fixed a problem where the ToInteger operator assumed all input strings
were
hexadecimal. The operator now handles both decimal strings and hex
strings
Fixed a problem where the ToInteger operator assumed all input strings were
hexadecimal. The operator now handles both decimal strings and hex strings
(prefixed with "0x").

Fixed a problem where the string length in the string object created as a
result of the internal ConvertToString procedure could be incorrect. This
potentially affected all implicit conversions and also the
ToDecimalString
potentially affected all implicit conversions and also the ToDecimalString
and ToHexString operators.

Fixed two problems in the ToString operator. If the length parameter was
zero, an incorrect string object was created and the value of the input
length parameter was inadvertently changed from zero to Ones.

Fixed a problem where the optional ResourceSource string in the
ExtendedIRQ
Fixed a problem where the optional ResourceSource string in the ExtendedIRQ
resource macro was ignored.

Simplified the interfaces to the internal division functions, reducing
code
Simplified the interfaces to the internal division functions, reducing code
size and complexity.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.9K Code,  11.4K Data,   89.3K Total
    Debug Version:     164.5K Code,  68.3K Data,  232.8K Total
  Current Release:
    Non-Debug Version:  77.8K Code,  11.5K Data,   89.3K Total
    Debug Version:     164.6K Code,  68.5K Data,  233.1K Total
```

2) iASL Compiler/Disassembler:

Implemented support for the ACPI 3.0 Timer operator.

Fixed a problem where the Default() operator was inadvertently ignored in
a
Fixed a problem where the Default() operator was inadvertently ignored in a
Switch/Case block. This was a problem in the translation of the Switch
statement to If...Else pairs.

Added support to allow a standalone Return operator, with no parentheses

(or
Added support to allow a standalone Return operator, with no parentheses (or
operands).

Fixed a problem with code generation for the ElseIf operator where the
translated Else...If parse tree was improperly constructed leading to the
loss of some code.

---
22 September 2004. Summary of changes for version 20040922:

1) ACPI CA Core Subsystem:

Fixed a problem with the implementation of the LNot() operator where
"Ones"
was not returned for the TRUE case. Changed the code to return Ones
instead
of (!Arg) which was usually 1. This change affects iASL constant folding
for
Fixed a problem with the implementation of the LNot() operator where "Ones"
was not returned for the TRUE case. Changed the code to return Ones instead
of (!Arg) which was usually 1. This change affects iASL constant folding for
this operator also.

Fixed a problem in AcpiUtInitializeBuffer where an existing buffer was
not
Fixed a problem in AcpiUtInitializeBuffer where an existing buffer was not
initialized properly -- Now zero the entire buffer in this case where the
buffer already exists.

Changed the interface to AcpiOsSleep from (UINT32 Seconds, UINT32
Milliseconds) to simply (ACPI_INTEGER Milliseconds). This simplifies all
related code considerably. This will require changes/updates to all OS
interface layers (OSLs.)

Implemented a new external interface, AcpiInstallExceptionHandler, to
allow
a system exception handler to be installed. This handler is invoked upon
any
Implemented a new external interface, AcpiInstallExceptionHandler, to allow
a system exception handler to be installed. This handler is invoked upon any
run-time exception that occurs during control method execution.

Added support for the DSDT in AcpiTbFindTable. This allows the
DataTableRegion() operator to access the local copy of the DSDT.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.8K Code,  11.4K Data,   89.2K Total
    Debug Version:     164.2K Code,  68.2K Data,  232.4K Total
  Current Release:
    Non-Debug Version:  77.9K Code,  11.4K Data,   89.3K Total
    Debug Version:     164.5K Code,  68.3K Data,  232.8K Total
```


2) iASL Compiler/Disassembler:

Fixed a problem with constant folding and the LNot operator. LNot was
returning 1 in the TRUE case, not Ones as per the ACPI specification.
This
returning 1 in the TRUE case, not Ones as per the ACPI specification. This
could result in the generation of an incorrect folded/reduced constant.

End-Of-File is now allowed within a "//"-style comment. A parse error no
longer occurs if such a comment is at the very end of the input ASL
source
longer occurs if such a comment is at the very end of the input ASL source
file.

Implemented the "-r" option to override the Revision in the table header.
The initial use of this option will be to simplify the evaluation of the
AML
interpreter by allowing a single ASL source module to be compiled for
either
The initial use of this option will be to simplify the evaluation of the AML
interpreter by allowing a single ASL source module to be compiled for either
32-bit or 64-bit integers.

---
27 August 2004. Summary of changes for version 20040827:

1) ACPI CA Core Subsystem:

- Implemented support for implicit object conversion in the non-numeric
logical operators (LEqual, LGreater, LGreaterEqual, LLess, LLessEqual,
and
logical operators (LEqual, LGreater, LGreaterEqual, LLess, LLessEqual, and
LNotEqual.) Any combination of Integers/Strings/Buffers may now be used;
the second operand is implicitly converted on the fly to match the type
of
the second operand is implicitly converted on the fly to match the type of
the first operand. For example:

```
    LEqual (Source1, Source2)
```

Source1 and Source2 must each evaluate to an integer, a string, or a
buffer.
The data type of Source1 dictates the required type of Source2. Source2
is
Source1 and Source2 must each evaluate to an integer, a string, or a buffer.
The data type of Source1 dictates the required type of Source2. Source2 is
implicitly converted if necessary to match the type of Source1.

- Updated and corrected the behavior of the string conversion support.
The
- Updated and corrected the behavior of the string conversion support. The
rules concerning conversion of buffers to strings (according to the ACPI
specification) are as follows:

ToDecimalString - explicit byte-wise conversion of buffer to string of
decimal values (0-255) separated by commas. ToHexString - explicit byte-
wise
decimal values (0-255) separated by commas. ToHexString - explicit byte-wise
conversion of buffer to string of hex values (0-FF) separated by commas.
ToString - explicit byte-wise conversion of buffer to string. Byte-by-
byte
copy with no transform except NULL terminated. Any other implicit buffer-
to-

string conversion - byte-wise conversion of buffer to string of hex values
ToString - explicit byte-wise conversion of buffer to string. Byte-by-byte copy with no transform except NULL terminated. Any other implicit buffer-to-string conversion - byte-wise conversion of buffer to string of hex values (0-FF) separated by spaces.

- Fixed typo in definition of AcpiGbl_EnableInterpreterSlack.

- Fixed a problem in AcpiNsGetPathnameLength where the returned length was
- Fixed a problem in AcpiNsGetPathnameLength where the returned length was one byte too short in the case of a node in the root scope. This could cause a fault during debug output.

- Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced
- Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the **debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and**
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.9K Code,  11.5K Data,   89.4K Total
    Debug Version:     164.1K Code,  68.3K Data,  232.4K Total
  Current Release:
    Non-Debug Version:  77.8K Code,  11.4K Data,   89.2K Total
    Debug Version:     164.2K Code,  68.2K Data,  232.4K Total
```

2) iASL Compiler/Disassembler:

- Fixed a Linux generation error.

---
16 August 2004. Summary of changes for version 20040816:

1) ACPI CA Core Subsystem:

Designed and implemented support within the AML interpreter for the so-**called "implicit return". This support returns the result of the last ASL**
called "implicit return". This support returns the result of the last ASL operation within a control method, in the absence of an explicit Return() operator. A few machines depend on this behavior, even though it is not **explicitly supported by the ASL language. It is optional support that can**
explicitly supported by the ASL language. It is optional support that can be enabled at runtime via the AcpiGbl_EnableInterpreterSlack flag.

Removed support for the PCI_Config address space from the internal low level
Removed support for the PCI_Config address space from the internal low level hardware interfaces (AcpiHwLowLevelRead and AcpiHwLowLevelWrite). This **support was not used internally, and would not work correctly anyway because**

support was not used internally, and would not work correctly anyway because the PCI bus number and segment number were not supported. There are **separate interfaces for PCI configuration space access because of the unique**
separate interfaces for PCI configuration space access because of the unique interface.

Code and Data Size: Current and previous core subsystem library sizes are **shown below. These are the code and data sizes for the acpica.lib produced**
shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the **debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and**
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  78.0K Code,  11.5K Data,   89.5K Total
    Debug Version:     164.1K Code,  68.2K Data,  232.3K Total
  Current Release:
    Non-Debug Version:  77.9K Code,  11.5K Data,   89.4K Total
    Debug Version:     164.1K Code,  68.3K Data,  232.4K Total
```

2) iASL Compiler/Disassembler:

Fixed a problem where constants in ASL expressions at the root level (not within a control method) could be inadvertently truncated during code generation. This problem was introduced in the 20040715 release.

---
15 July 2004. Summary of changes for version 20040715:

1) ACPI CA Core Subsystem:

Restructured the internal HW GPE interfaces to pass/track the current state
Restructured the internal HW GPE interfaces to pass/track the current state of interrupts (enabled/disabled) in order to avoid possible deadlock and increase flexibility of the interfaces.

Implemented a "lexicographical compare" for String and Buffer objects within the logical operators -- LGreater, LLess, LGreaterEqual, and LLessEqual - as per further clarification to the ACPI specification. Behavior is similar
Implemented a "lexicographical compare" for String and Buffer objects within the logical operators -- LGreater, LLess, LGreaterEqual, and LLessEqual -- as per further clarification to the ACPI specification. Behavior is similar to C library "strcmp".

Completed a major reduction in CPU stack use for the AcpiGetFirmwareTable external function. In the 32-bit non-debug case, the stack use has been reduced from 168 bytes to 32 bytes.

Deployed a new run-time configuration flag, AcpiGbl_EnableInterpreterSlack,
Deployed a new run-time configuration flag, AcpiGbl_EnableInterpreterSlack, whose purpose is to allow the AML interpreter to forgive certain bad AML

constructs. Default setting is FALSE.

Implemented the first use of AcpiGbl_EnableInterpreterSlack in the Field IO
support code. If enabled, it allows field access to go beyond the end of a
region definition if the field is within the region length rounded up to the
Implemented the first use of AcpiGbl_EnableInterpreterSlack in the Field IO support code. If enabled, it allows field access to go beyond the end of a region definition if the field is within the region length rounded up to the
next access width boundary (a common coding error.)

Renamed OSD_HANDLER to ACPI_OSD_HANDLER, and OSD_EXECUTION_CALLBACK to
ACPI_OSD_EXEC_CALLBACK for consistency with other ACPI symbols. Also, these
ACPI_OSD_EXEC_CALLBACK for consistency with other ACPI symbols. Also, these
symbols are lowercased by the latest version of the AcpiSrc tool.

The prototypes for the PCI interfaces in acpiosxf.h have been updated to
rename "Register" to simply "Reg" to prevent certain compilers from
complaining.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.8K Code,  11.5K Data,   89.3K Total
    Debug Version:     163.8K Code,  68.2K Data,  232.0K Total
  Current Release:
    Non-Debug Version:  78.0K Code,  11.5K Data,   89.5K Total
    Debug Version:     164.1K Code,  68.2K Data,  232.3K Total
```

2) iASL Compiler/Disassembler:

Implemented full support for Package objects within the Case() operator.
Note: The Break() operator is currently not supported within Case blocks
(TermLists) as there is some question about backward compatibility with ACPI
(TermLists) as there is some question about backward compatibility with ACPI
1.0 interpreters.

Fixed a problem where complex terms were not supported properly within the
Fixed a problem where complex terms were not supported properly within the
Switch() operator.

Eliminated extraneous warning for compiler-emitted reserved names of the
form "_T_x". (Used in Switch/Case operators.)

Eliminated optimization messages for "_T_x" objects and small constants
within the DefinitionBlock operator.

---
15 June 2004. Summary of changes for version 20040615:

1) ACPI CA Core Subsystem:

Implemented support for Buffer and String objects (as per ACPI 2.0) for the
Implemented support for Buffer and String objects (as per ACPI 2.0) for the
following ASL operators: LEqual, LGreater, LLess, LGreaterEqual, and
LLessEqual.

All directory names in the entire source package are lower case, as they
were in earlier releases.

Implemented "Disassemble" command in the AML debugger that will disassemble
Implemented "Disassemble" command in the AML debugger that will disassemble
a single control method.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.7K Code,  11.5K Data,   89.2K Total
    Debug Version:     163.3K Code,  67.2K Data,  230.5K Total

  Current Release:
    Non-Debug Version:  77.8K Code,  11.5K Data,   89.3K Total
    Debug Version:     163.8K Code,  68.2K Data,  232.0K Total
```

2) iASL Compiler/Disassembler:

Implemented support for Buffer and String objects (as per ACPI 2.0) for the
Implemented support for Buffer and String objects (as per ACPI 2.0) for the
following ASL operators: LEqual, LGreater, LLess, LGreaterEqual, and
LLessEqual.

All directory names in the entire source package are lower case, as they
were in earlier releases.

Fixed a fault when using the -g or -d<nofilename> options if the FADT was
not found.

Fixed an issue with the Windows version of the compiler where later versions
Fixed an issue with the Windows version of the compiler where later versions
of Windows place the FADT in the registry under the name "FADT" and not
"FACP" as earlier versions did. This applies when using the -g or -
d<nofilename> options. The compiler now looks for both strings as
necessary.

Fixed a problem with compiler namepath optimization where a namepath

within
the Scope() operator could not be optimized if the namepath was a subpath
of
Fixed a problem with compiler namepath optimization where a namepath within
the Scope() operator could not be optimized if the namepath was a subpath of
the current scope path.

---
27 May 2004. Summary of changes for version 20040527:

1) ACPI CA Core Subsystem:

Completed a new design and implementation for EBDA (Extended BIOS Data
Area)
support in the RSDP scan code. The original code improperly scanned for
the
EBDA by simply scanning from memory location 0 to 0x400. The correct
method
Completed a new design and implementation for EBDA (Extended BIOS Data Area)
support in the RSDP scan code. The original code improperly scanned for the
EBDA by simply scanning from memory location 0 to 0x400. The correct method
is to first obtain the EBDA pointer from within the BIOS data area, then
scan 1K of memory starting at the EBDA pointer. There appear to be few
if
scan 1K of memory starting at the EBDA pointer. There appear to be few if
any machines that place the RSDP in the EBDA, however.

Integrated a fix for a possible fault during evaluation of BufferField
arguments. Obsolete code that was causing the problem was removed.

Found and fixed a problem in the Field Support Code where data could be
corrupted on a bit field read that starts on an aligned boundary but does
not end on an aligned boundary. Merged the read/write "datum length"
calculation code into a common procedure.

Rolled in a couple of changes to the FreeBSD-specific header.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.6K Code,  11.5K Data,   89.1K Total
    Debug Version:     163.2K Code,  67.2K Data,  230.4K Total
  Current Release:
    Non-Debug Version:  77.7K Code,  11.5K Data,   89.2K Total
    Debug Version:     163.3K Code,  67.2K Data,  230.5K Total
```

2) iASL Compiler/Disassembler:

Fixed a generation warning produced by some overly-verbose compilers for
a
Fixed a generation warning produced by some overly-verbose compilers for a
64-bit constant.

---
14 May 2004. Summary of changes for version 20040514:

1) ACPI CA Core Subsystem:

Fixed a problem where hardware GPE enable bits sometimes not set properly
during and after GPE method execution. Result of 04/27 changes.

Removed extra "clear all GPEs" when sleeping/waking.

Removed AcpiHwEnableGpe and AcpiHwDisableGpe, replaced by the single
AcpiHwWriteGpeEnableReg. Changed a couple of calls to the functions above
to
AcpiHwWriteGpeEnableReg. Changed a couple of calls to the functions above to
the new AcpiEv* calls as appropriate.

ACPI_OS_NAME was removed from the OS-specific headers. The default name
is
now "Microsoft Windows NT" for maximum compatibility. However this can
be
ACPI_OS_NAME was removed from the OS-specific headers. The default name is
now "Microsoft Windows NT" for maximum compatibility. However this can be
changed by modifying the acconfig.h file.

Allow a single invocation of AcpiInstallNotifyHandler for a handler that
traps both types of notifies (System, Device). Use ACPI_ALL_NOTIFY flag.

Run _INI methods on ThermalZone objects. This is against the ACPI
specification, but there is apparently ASL code in the field that has
these
specification, but there is apparently ASL code in the field that has these
_INI methods, and apparently "other" AML interpreters execute them.

Performed a full 16/32/64 bit lint that resulted in some small changes.

Added a sleep simulation command to the AML debugger to test sleep code.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
  Previous Release:
    Non-Debug Version:  77.6K Code,  11.5K Data,   89.1K Total
    Debug Version:     162.9K Code,  67.0K Data,  229.9K Total
  Current Release:
    Non-Debug Version:  77.6K Code,  11.5K Data,   89.1K Total
    Debug Version:     163.2K Code,  67.2K Data,  230.4K Total
```

---
27 April 2004. Summary of changes for version 20040427:

1) ACPI CA Core Subsystem:

Completed a major overhaul of the GPE handling within ACPI CA. There are
now three types of GPEs: wake-only, runtime-only, and combination

wake/run.
now three types of GPEs: wake-only, runtime-only, and combination wake/run.
The only GPEs allowed to be combination wake/run are for button-style
devices such as a control-method power button, control-method sleep
button,
or a notebook lid switch. GPEs that have an _Lxx or _Exx method and are
not
devices such as a control-method power button, control-method sleep button,
or a notebook lid switch. GPEs that have an _Lxx or _Exx method and are not
referenced by any _PRW methods are marked for "runtime" and hardware
enabled. Any GPE that is referenced by a _PRW method is marked for
"wake"
enabled. Any GPE that is referenced by a _PRW method is marked for "wake"
(and disabled at runtime). However, at sleep time, only those GPEs that
have been specifically enabled for wake via the AcpiEnableGpe interface
will
have been specifically enabled for wake via the AcpiEnableGpe interface will
actually be hardware enabled.

A new external interface has been added, AcpiSetGpeType(), that is meant
to
be used by device drivers to force a GPE to a particular type. It will
be
A new external interface has been added, AcpiSetGpeType(), that is meant to
be used by device drivers to force a GPE to a particular type. It will be
especially useful for the drivers for the button devices mentioned above.

Completed restructuring of the ACPI CA initialization sequence so that
default operation region handlers are installed before GPEs are
initialized
and the _PRW methods are executed. This will prevent errors when the
_PRW
default operation region handlers are installed before GPEs are initialized
and the _PRW methods are executed. This will prevent errors when the _PRW
methods attempt to access system memory or I/O space.

GPE enable/disable no longer reads the GPE enable register. We now keep
the
GPE enable/disable no longer reads the GPE enable register. We now keep the
enable info for runtime and wake separate and in the GPE_EVENT_INFO. We
thus no longer depend on the hardware to maintain these bits.

Always clear the wake status and fixed/GPE status bits before sleep, even
for state S5.

Improved the AML debugger output for displaying the GPE blocks and their
current status.

Added new strings for the _OSI method, of the form "Windows 2001 SPx"
where
Added new strings for the _OSI method, of the form "Windows 2001 SPx" where
x = 0,1,2,3,4.

Fixed a problem where the physical address was incorrectly calculated
when
the Load() operator was used to directly load from an Operation Region
(vs.
loading from a Field object.) Also added check for minimum table length
for
Fixed a problem where the physical address was incorrectly calculated when
the Load() operator was used to directly load from an Operation Region (vs.
loading from a Field object.) Also added check for minimum table length for
this case.

Fix for multiple mutex acquisition. Restore original thread SyncLevel on
mutex release.

Added ACPI_VALID_SXDS flag to the AcpiGetObjectInfo interface for
consistency with the other fields returned.

Shrunk the ACPI_GPE_EVENT_INFO structure by 40%. There is one such
structure for each GPE in the system, so the size of this structure is
important.

CPU stack requirement reduction: Cleaned up the method execution and
object
CPU stack requirement reduction: Cleaned up the method execution and object
evaluation paths so that now a parameter structure is passed, instead of
copying the various method parameters over and over again.

In evregion.c: Correctly exit and reenter the interpreter region if and
only if dispatching an operation region request to a user-installed
handler.
only if dispatching an operation region request to a user-installed handler.
Do not exit/reenter when dispatching to a default handler (e.g., default
system memory or I/O handlers)

Notes for updating drivers for the new GPE support. The following
changes
must be made to ACPI-related device drivers that are attached to one or
more
GPEs: (This information will be added to the ACPI CA Programmer
Reference.)
Notes for updating drivers for the new GPE support. The following changes
must be made to ACPI-related device drivers that are attached to one or more
GPEs: (This information will be added to the ACPI CA Programmer Reference.)

1) AcpiInstallGpeHandler no longer automatically enables the GPE, you
must
1) AcpiInstallGpeHandler no longer automatically enables the GPE, you must
explicitly call AcpiEnableGpe.
2) There is a new interface called AcpiSetGpeType. This should be called
before enabling the GPE. Also, this interface will automatically disable
the GPE if it is currently enabled.
3) AcpiEnableGpe no longer supports a GPE type flag.

Specific drivers that must be changed:
1) EC driver:
    AcpiInstallGpeHandler (NULL, GpeNum, ACPI_GPE_EDGE_TRIGGERED,
AeGpeHandler, NULL);
    AcpiSetGpeType (NULL, GpeNum, ACPI_GPE_TYPE_RUNTIME);
    AcpiEnableGpe (NULL, GpeNum, ACPI_NOT_ISR);

2) Button Drivers (Power, Lid, Sleep):
Run _PRW method under parent device
If _PRW exists: /* This is a control-method button */
    Extract GPE number and possibly GpeDevice
    AcpiSetGpeType (GpeDevice, GpeNum, ACPI_GPE_TYPE_WAKE_RUN);
    AcpiEnableGpe (GpeDevice, GpeNum, ACPI_NOT_ISR);

For all other devices that have _PRWs, we automatically set the GPE type
to
ACPI_GPE_TYPE_WAKE, but the GPE is NOT automatically (wake) enabled.
This
must be done on a selective basis, usually requiring some kind of user
app
For all other devices that have _PRWs, we automatically set the GPE type to
ACPI_GPE_TYPE_WAKE, but the GPE is NOT automatically (wake) enabled. This
must be done on a selective basis, usually requiring some kind of user app
to allow the user to pick the wake devices.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
   Previous Release:
     Non-Debug Version:  77.0K Code,  11.4K Data,   88.4K Total
     Debug Version:     161.0K Code,  66.3K Data,  227.3K Total
   Current Release:

     Non-Debug Version:  77.6K Code,  11.5K Data,   89.1K Total
     Debug Version:     162.9K Code,  67.0K Data,  229.9K Total
```

---

02 April 2004. Summary of changes for version 20040402:

1) ACPI CA Core Subsystem:

Fixed an interpreter problem where an indirect store through an ArgX
parameter was incorrectly applying the "implicit conversion rules" during
the store. From the ACPI specification: "If the target is a method local
or
the store. From the ACPI specification: "If the target is a method local or
argument (LocalX or ArgX), no conversion is performed and the result is
stored directly to the target". The new behavior is to disable implicit
conversion during ALL stores to an ArgX.

Changed the behavior of the _PRW method scan to ignore any and all errors
returned by a given _PRW. This prevents the scan from aborting from the
failure of any single _PRW.

Moved the runtime configuration parameters from the global init procedure
to
Moved the runtime configuration parameters from the global init procedure to
static variables in acglobal.h. This will allow the host to override the
default values easily.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

```
   Previous Release:
     Non-Debug Version:  76.9K Code,  11.4K Data,   88.3K Total
     Debug Version:     160.8K Code,  66.1K Data,  226.9K Total
```



```
   Current Release:
     Non-Debug Version:  77.0K Code,  11.4K Data,   88.4K Total
     Debug Version:     161.0K Code,  66.3K Data,  227.3K Total
```

2) iASL Compiler/Disassembler:

iASL now fully disassembles SSDTs. However, External() statements are
not
iASL now fully disassembles SSDTs. However, External() statements are not
generated automatically for unresolved symbols at this time. This is a
planned feature for future implementation.

Fixed a scoping problem in the disassembler that occurs when the type of
the
Fixed a scoping problem in the disassembler that occurs when the type of the
target of a Scope() operator is overridden. This problem caused an
incorrectly nested internal namespace to be constructed.

Any warnings or errors that are emitted during disassembly are now
commented
out automatically so that the resulting file can be recompiled without
any
Any warnings or errors that are emitted during disassembly are now commented
out automatically so that the resulting file can be recompiled without any
hand editing.

---

26 March 2004. Summary of changes for version 20040326:

1) ACPI CA Core Subsystem:

Implemented support for "wake" GPEs via interaction between GPEs and the
_PRW methods. Every GPE that is pointed to by one or more _PRWs is
identified as a WAKE GPE and by default will no longer be enabled at
runtime. Previously, we were blindly enabling all GPEs with a
corresponding
_Lxx or _Exx method - but most of these turn out to be WAKE GPEs anyway.
We
runtime. Previously, we were blindly enabling all GPEs with a corresponding
_Lxx or _Exx method - but most of these turn out to be WAKE GPEs anyway. We
believe this has been the cause of thousands of "spurious" GPEs on some
systems.

This new GPE behavior is can be reverted to the original behavior (enable
ALL GPEs at runtime) via a runtime flag.

Fixed a problem where aliased control methods could not access objects
properly. The proper scope within the namespace was not initialized
(transferred to the target of the aliased method) before executing the
target method.

Fixed a potential race condition on internal object deletion on the
return
Fixed a potential race condition on internal object deletion on the return
object in AcpiEvaluateObject.

Integrated a fix for resource descriptors where both _MEM and _MTP were
being extracted instead of just _MEM. (i.e. bitmask was incorrectly too
wide, 0x0F instead of 0x03.)

Added a special case for ACPI_ROOT_OBJECT in AcpiUtGetNodeName,
preventing
a
Added a special case for ACPI_ROOT_OBJECT in AcpiUtGetNodeName, preventing a
fault in some cases.

Updated Notify() values for debug statements in evmisc.c

Return proper status from AcpiUtMutexInitialize, not just simply AE_OK.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

  Previous Release:

    Non-Debug Version: 76.5K Code, 11.3K Data,   87.8K Total
    Debug Version: 160.3K Code, 66.0K Data, 226.3K Total
  Current Release:
    Non-Debug Version: 76.9K Code, 11.4K Data,   88.3K Total
    Debug Version: 160.8K Code, 66.1K Data, 226.9K Total

----
11 March 2004. Summary of changes for version 20040311:

1) ACPI CA Core Subsystem:

Fixed a problem where errors occurring during the parse phase of control
method execution did not abort cleanly. For example, objects created and
installed in the namespace were not deleted. This caused all subsequent
invocations of the method to return the AE_ALREADY_EXISTS exception.

Implemented a mechanism to force a control method to "Serialized"
execution
Implemented a mechanism to force a control method to "Serialized" execution
if the method attempts to create namespace objects. (The root of the
AE_ALREADY_EXISTS problem.)

Implemented support for the predefined _OSI "internal" control method.
Initial supported strings are "Linux", "Windows 2000", "Windows 2001",
and
"Windows 2001.1", and can be easily upgraded for new strings as
necessary.
Initial supported strings are "Linux", "Windows 2000", "Windows 2001", and
"Windows 2001.1", and can be easily upgraded for new strings as necessary.
This feature will allow "other" operating systems to execute the fully
tested, "Windows" code path through the ASL code

Global Lock Support: Now allows multiple acquires and releases with any
internal thread. Removed concept of "owning thread" for this special
mutex.
internal thread. Removed concept of "owning thread" for this special mutex.

Fixed two functions that were inappropriately declaring large objects on
the
CPU stack: PsParseLoop, NsEvaluateRelative. Reduces the stack usage
during
Fixed two functions that were inappropriately declaring large objects on the
CPU stack: PsParseLoop, NsEvaluateRelative. Reduces the stack usage during
method execution considerably.

Fixed a problem in the ACPI 2.0 FACS descriptor (actbl2.h) where the
S4Bios_f field was incorrectly defined as UINT32 instead of UINT32_BIT.

Fixed a problem where AcpiEvGpeDetect would fault if there were no GPEs
defined on the machine.

Implemented two runtime options: One to force all control method
execution
to "Serialized" to mimic Windows behavior, another to disable _OSI
support
Implemented two runtime options: One to force all control method execution
to "Serialized" to mimic Windows behavior, another to disable _OSI support
if it causes problems on a given machine.

Code and Data Size: Current and previous core subsystem library sizes are
shown below. These are the code and data sizes for the acpica.lib
produced
shown below. These are the code and data sizes for the acpica.lib produced
by the Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code includes the
debug output trace mechanism and has a much larger code and data size.
Note
that these values will vary depending on the efficiency of the compiler
and
debug output trace mechanism and has a much larger code and data size. Note
that these values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

  Previous Release:
    Non-Debug Version: 74.8K Code, 10.1K Data,   84.9K Total
    Debug Version: 158.7K Code, 65.1K Data, 223.8K Total
  Current Release:
    Non-Debug Version: 76.5K Code, 11.3K Data,   87.8K Total
    Debug Version: 160.3K Code, 66.0K Data, 226.3K Total

2) iASL Compiler/Disassembler:

Fixed an array size problem for FreeBSD that would cause the compiler to
fault.

----
20 February 2004. Summary of changes for version 20040220:

1) ACPI CA Core Subsystem:

Implemented execution of _SxD methods for Device objects in the
GetObjectInfo interface.

Fixed calls to _SST method to pass the correct arguments.

Added a call to _SST on wake to restore to "working" state.

Check for End-Of-Buffer failure case in the WalkResources interface.

Integrated fix for 64-bit alignment issue in acglobal.h by moving two
structures to the beginning of the file.

After wake, clear GPE status register(s) before enabling GPEs.

After wake, clear/enable power button. (Perhaps we should clear/enable
all
After wake, clear/enable power button. (Perhaps we should clear/enable all
fixed events upon wake.)

Fixed a couple of possible memory leaks in the Namespace manager.

Integrated latest acnetbsd.h file.

---
11 February 2004. Summary of changes for version 20040211:

1) ACPI CA Core Subsystem:

Completed investigation and implementation of the call-by-reference
mechanism for control method arguments.

Fixed a problem where a store of an object into an indexed package could
fail if the store occurs within a different method than the method that
created the package.

**Fixed a problem where the ToDecimal operator could return incorrect
results.**
Fixed a problem where the ToDecimal operator could return incorrect results.

**Fixed a problem where the CopyObject operator could fail on some of the
more**
Fixed a problem where the CopyObject operator could fail on some of the more
obscure objects (e.g., Reference objects.)

Improved the output of the Debug object to display buffer, package, and
index objects.

**Fixed a problem where constructs of the form "RefOf (ArgX)" did not
return**
Fixed a problem where constructs of the form "RefOf (ArgX)" did not return
the expected result.

Added permanent ACPI_REPORT_ERROR macros for all instances of the
ACPI_AML_INTERNAL exception.

Integrated latest version of acfreebsd.h

---
16 January 2004. Summary of changes for version 20040116:

The purpose of this release is primarily to update the copyright years in
each module, thus causing a huge number of diffs. There are a few small
functional changes, however.

1) ACPI CA Core Subsystem:

**Improved error messages when there is a problem finding one or more of
the**
Improved error messages when there is a problem finding one or more of the
required base ACPI tables

Reintroduced the definition of APIC_HEADER in actbl.h

Changed definition of MADT_ADDRESS_OVERRIDE to 64 bits (actbl.h)

Removed extraneous reference to NewObj in dsmthdat.c

2) iASL compiler

**Fixed a problem introduced in December that disabled the correct
disassembly**
Fixed a problem introduced in December that disabled the correct disassembly
of Resource Templates

---
03 December 2003. Summary of changes for version 20031203:

1) ACPI CA Core Subsystem:

Changed the initialization of Operation Regions during subsystem
init to perform two entire walks of the ACPI namespace; The first
to initialize the regions themselves, the second to execute the
_REG methods. This fixed some interdependencies across _REG
methods found on some machines.

Fixed a problem where a Store(Local0, Local1) could simply update
the object reference count, and not create a new copy of the
object if the Local1 is uninitialized.

Implemented support for the _SST reserved method during sleep
transitions.

Implemented support to clear the SLP_TYP and SLP_EN bits when
waking up, this is apparently required by some machines.

When sleeping, clear the wake status only if SleepState is not S5.

Fixed a problem in AcpiRsExtendedIrqResource() where an incorrect
pointer arithmetic advanced a string pointer too far.

Fixed a problem in AcpiTbGetTablePtr() where a garbage pointer
could be returned if the requested table has not been loaded.

Within the support for IRQ resources, restructured the handling of
the active and edge/level bits.

Fixed a few problems in AcpiPsxExecute() where memory could be
leaked under certain error conditions.

Improved error messages for the cases where the ACPI mode could
not be entered.

Code and Data Size: Current and previous core subsystem library
sizes are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and
these values do not include any ACPI driver or OSPM code. The
debug version of the code includes the debug output trace
mechanism and has a much larger code and data size. Note that
these values will vary depending on the efficiency of the compiler
and the compiler options used during generation.

```
  Previous Release (20031029):
    Non-Debug Version:  74.4K Code,  10.1K Data,   84.5K Total
    Debug Version:     158.3K Code,  65.0K Data,  223.3K Total
  Current Release:
    Non-Debug Version:  74.8K Code,  10.1K Data,   84.9K Total
    Debug Version:     158.7K Code,  65.1K Data,  223.8K Total
```

2) iASL Compiler/Disassembler:

Implemented a fix for the iASL disassembler where a bad index was
generated. This was most noticeable on 64-bit platforms

---
29 October 2003. Summary of changes for version 20031029:

1) ACPI CA Core Subsystem:

Fixed a problem where a level-triggered GPE with an associated
_Lxx control method was incorrectly cleared twice.

Fixed a problem with the Field support code where an access can
occur beyond the end-of-region if the field is non-aligned but
extends to the very end of the parent region (resulted in an
AE_AML_REGION_LIMIT exception.)

Fixed a problem with ACPI Fixed Events where an RT Clock handler
would not get invoked on an RTC event. The RTC event bitmasks for
the PM1 registers were not being initialized properly.

Implemented support for executing _STA and _INI methods for
Processor objects. Although this is currently not part of the
ACPI specification, there is existing ASL code that depends on the
init-time execution of these methods.

Implemented and deployed a GetDescriptorName function to decode
the various types of internal descriptors. Guards against null
descriptors during debug output also.

Implemented and deployed a GetNodeName function to extract the 4-
character namespace node name. This function simplifies the debug
and error output, as well as guarding against null pointers during
output.

Implemented and deployed the ACPI_FORMAT_UINT64 helper macro to
simplify the debug and error output of 64-bit integers. This
macro replaces the HIDWORD and LODWORD macros for dumping these
integers.

Updated the implementation of the Stall() operator to only call
AcpiOsStall(), and also return an error if the operand is larger
than 255. This preserves the required behavior of not
relinquishing the processor, as would happen if AcpiOsSleep() was
called for "long stalls".

Constructs of the form "Store(LocalX,LocalX)" where LocalX is not
initialized are now treated as NOOPs.

Cleaned up a handful of warnings during 64-bit generation.

Fixed a reported error where and incorrect GPE number was passed
to the GPE dispatch handler. This value is only used for error
output, however. Used this opportunity to clean up and streamline
the GPE dispatch code.

Code and Data Size: Current and previous core subsystem library
sizes are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and
these values do not include any ACPI driver or OSPM code. The

debug version of the code includes the debug output trace
mechanism and has a much larger code and data size. Note that
these values will vary depending on the efficiency of the compiler
and the compiler options used during generation.

```
  Previous Release (20031002):
    Non-Debug Version:  74.1K Code,   9.7K Data,   83.8K Total
    Debug Version:     157.9K Code,  64.8K Data,  222.7K Total
  Current Release:
    Non-Debug Version:  74.4K Code,  10.1K Data,   84.5K Total
    Debug Version:     158.3K Code,  65.0K Data,  223.3K Total
```

2) iASL Compiler/Disassembler:

Updated the iASL compiler to return an error if the operand to the
Stall() operator is larger than 255.

---

02 October 2003. Summary of changes for version 20031002:

1) ACPI CA Core Subsystem:

Fixed a problem with Index Fields where the index was not
incremented for fields that require multiple writes to the
index/data registers (Fields that are wider than the data
register.)

Fixed a problem with all Field objects where a write could go
beyond the end-of-field if the field was larger than the access
granularity and therefore required multiple writes to complete the
request. An extra write beyond the end of the field could happen
inadvertently.

Fixed a problem with Index Fields where a BUFFER_OVERFLOW error
would incorrectly be returned if the width of the Data Register
was larger than the specified field access width.

Completed fixes for LoadTable() and Unload() and verified their
operation. Implemented full support for the "DdbHandle" object
throughout the ACPI CA subsystem.

Implemented full support for the MADT and ECDT tables in the ACPI
CA header files. Even though these tables are not directly
consumed by ACPI CA, the header definitions are useful for ACPI
device drivers.

Integrated resource descriptor fixes posted to the Linux ACPI
list. This included checks for minimum descriptor length, and
support for trailing NULL strings within descriptors that have
optional string elements.

Code and Data Size: Current and previous core subsystem library
sizes are shown below. These are the code and data sizes for the
acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and
these values do not include any ACPI driver or OSPM code. The
debug version of the code includes the debug output trace
mechanism and has a much larger code and data size. Note that
these values will vary depending on the efficiency of the compiler
and the compiler options used during generation.

```
  Previous Release (20030918):
    Non-Debug Version:  73.9K Code,   9.7K Data,   83.6K Total
    Debug Version:     157.3K Code,  64.5K Data,  221.8K Total
  Current Release:
    Non-Debug Version:  74.1K Code,   9.7K Data,   83.8K Total
    Debug Version:     157.9K Code,  64.8K Data,  222.7K Total
```

2) iASL Compiler:

Implemented detection of non-ASCII characters within the input
source ASL file. This catches attempts to compile binary (AML)
files early in the compile, with an informative error message.

Fixed a problem where the disassembler would fault if the output
filename could not be generated or if the output file could not be
opened.

```
10782 ----------------------------------------
10783 18 September 2003.  Summary of changes for version 20030918:


10786 1) ACPI CA Core Subsystem:

10788 Found and fixed a longstanding problem with the late execution of
10789 the various deferred AML opcodes (such as Operation Regions,
10790 Buffer Fields, Buffers, and Packages).  If the name string
10791 specified for the name of the new object placed the object in a
10792 scope other than the current scope, the initialization/execution
10793 of the opcode failed.  The solution to this problem was to
10794 implement a mechanism where the late execution of such opcodes
10795 does not attempt to lookup/create the name a second time in an
10796 incorrect scope.  This fixes the "region size computed
10797 incorrectly" problem.

10799 Fixed a call to AcpiHwRegisterWrite in hwregs.c that was causing a
10800 Global Lock AE_BAD_PARAMETER error.

10802 Fixed several 64-bit issues with prototypes, casting and data
10803 types.

10805 Removed duplicate prototype from acdisasm.h

10807 Fixed an issue involving EC Operation Region Detach (Shaohua Li)

10809 Code and Data Size: Current and previous core subsystem library
10810 sizes are shown below.  These are the code and data sizes for the
10811 acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and
10812 these values do not include any ACPI driver or OSPM code.  The
10813 debug version of the code includes the debug output trace
10814 mechanism and has a much larger code and data size.  Note that
10815 these values will vary depending on the efficiency of the compiler
10816 and the compiler options used during generation.

10818   Previous Release:

10820     Non-Debug Version:  73.7K Code,   9.7K Data,   83.4K Total
10821     Debug Version:     156.9K Code,  64.2K Data,  221.1K Total
10822   Current Release:
10823     Non-Debug Version:  73.9K Code,   9.7K Data,   83.6K Total
10824     Debug Version:     157.3K Code,  64.5K Data,  221.8K Total


10827 2) Linux:

10829 Fixed the AcpiOsSleep implementation in osunixxf.c to pass the
10830 correct sleep time in seconds.

10832 ----------------------------------------
10833 14 July 2003.  Summary of changes for version 20030619:

10835 1) ACPI CA Core Subsystem:

10837 Parse SSDTs in order discovered, as opposed to reverse order
10838 (Hrvoje Habjanic)

10840 Fixes from FreeBSD and NetBSD. (Frank van der Linden, Thomas
10841 Klausner,
10842    Nate Lawson)


10845 2) Linux:
```



```
10847 Dynamically allocate SDT list (suggested by Andi Kleen)

10849 proc function return value cleanups (Andi Kleen)

10851 Correctly handle NMI watchdog during long stalls (Andrew Morton)

10853 Make it so acpismp=force works (reported by Andrew Morton)


10856 ----------------------------------------
10857 19 June 2003.  Summary of changes for version 20030619:

10859 1) ACPI CA Core Subsystem:

10861 Fix To/FromBCD, eliminating the need for an arch-specific #define.

10863 Do not acquire a semaphore in the S5 shutdown path.

10865 Fix ex_digits_needed for 0. (Takayoshi Kochi)

10867 Fix sleep/stall code reversal. (Andi Kleen)

10869 Revert a change having to do with control method calling
10870 semantics.

10872 2) Linux:

10874 acpiphp update (Takayoshi Kochi)

10876 Export acpi_disabled for sonypi (Stelian Pop)

10878 Mention acpismp=force in config help

10880 Re-add acpitable.c and acpismp=force. This improves backwards

10882 compatibility and also cleans up the code to a significant degree.

10884 Add ASUS Value-add driver (Karol Kozimor and Julien Lerouge)

10886 ----------------------------------------
10887 22 May 2003.  Summary of changes for version 20030522:

10889 1) ACPI CA Core Subsystem:

10891 Found and fixed a reported problem where an AE_NOT_FOUND error
10892 occurred occasionally during _BST evaluation.  This turned out to
10893 be an Owner ID allocation issue where a called method did not get
10894 a new ID assigned to it.  Eventually, (after 64k calls), the Owner
10895 ID UINT16 would wraparound so that the ID would be the same as the
10896 caller's and the called method would delete the caller's
10897 namespace.

10899 Implemented extended error reporting for control methods that are
10900 aborted due to a run-time exception.  Output includes the exact
10901 AML instruction that caused the method abort, a dump of the method
10902 locals and arguments at the time of the abort, and a trace of all
10903 nested control method calls.

10905 Modified the interpreter to allow the creation of buffers of zero
10906 length from the AML code. Implemented new code to ensure that no
10907 attempt is made to actually allocate a memory buffer (of length
10908 zero) - instead, a simple buffer object with a NULL buffer pointer
10909 and length zero is created.  A warning is no longer issued when
10910 the AML attempts to create a zero-length buffer.

10912 Implemented a workaround for the "leading asterisk issue" in
```

_HIDs, _UIDs, and _CIDs in the AML interpreter. One leading asterisk is automatically removed if present in any HID, UID, or CID strings. The iASL compiler will still flag this asterisk as an error, however.

Implemented full support for _CID methods that return a package of multiple CIDs (Compatible IDs). The AcpiGetObjectInfo() interface now additionally returns a device _CID list if present. This required a change to the external interface in order to pass an ACPI_BUFFER object as a parameter since the _CID list is of variable length.

Fixed a problem with the new AE_SAME_HANDLER exception where handler initialization code did not know about this exception.

Code and Data Size: Current and previous core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release (20030509):
    Non-Debug Version:  73.4K Code,   9.7K Data,   83.1K Total
    Debug Version:     156.1K Code,  63.9K Data,  220.0K Total
  Current Release:
    Non-Debug Version:  73.7K Code,   9.7K Data,   83.4K Total
    Debug Version:     156.9K Code,  64.2K Data,  221.1K Total
```

2) Linux:

Fixed a bug in which we would reinitialize the ACPI interrupt after it was already working, thus disabling all ACPI and the IRQs for any other device sharing the interrupt. (Thanks to Stian Jordet)

Toshiba driver update (John Belmonte)

Return only 0 or 1 for our interrupt handler status (Andrew Morton)

3) iASL Compiler:

Fixed a reported problem where multiple (nested) ElseIf() statements were not handled correctly by the compiler, resulting in incorrect warnings and incorrect AML code. This was a problem in both the ASL parser and the code generator.

4) Documentation:

Added changes to existing interfaces, new exception codes, and new text concerning reference count object management versus garbage collection.

---

09 May 2003. Summary of changes for version 20030509.

1) ACPI CA Core Subsystem:

Changed the subsystem initialization sequence to hold off

installation of address space handlers until the hardware has been initialized and the system has entered ACPI mode. This is because the installation of space handlers can cause _REG methods to be run. Previously, the _REG methods could potentially be run before ACPI mode was enabled.

Fixed some memory leak issues related to address space handler and notify handler installation. There were some problems with the reference count mechanism caused by the fact that the handler objects are shared across several namespace objects.

Fixed a reported problem where reference counts within the namespace were not properly updated when named objects created by method execution were deleted.

Fixed a reported problem where multiple SSDTs caused a deletion issue during subsystem termination. Restructured the table data structures to simplify the linked lists and the related code.

Fixed a problem where the table ID associated with secondary tables (SSDTs) was not being propagated into the namespace objects created by those tables. This would only present a problem for tables that are unloaded at run-time, however.

Updated AcpiOsReadable and AcpiOsWritable to use the ACPI_SIZE type as the length parameter (instead of UINT32).

Solved a long-standing problem where an ALREADY_EXISTS error appears on various systems. This problem could happen when there are multiple PCI_Config operation regions under a single PCI root bus. This doesn't happen very frequently, but there are some systems that do this in the ASL.

Fixed a reported problem where the internal DeleteNode function was incorrectly handling the case where a namespace node was the first in the parent's child list, and had additional peers (not the only child, but first in the list of children.)

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a much larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release
    Non-Debug Version:  73.7K Code,   9.5K Data,   83.2K Total
    Debug Version:     156.1K Code,  63.6K Data,  219.7K Total
  Current Release:
    Non-Debug Version:  73.4K Code,   9.7K Data,   83.1K Total
    Debug Version:     156.1K Code,  63.9K Data,  220.0K Total
```

2) Linux:

Allow ":" in OS override string (Ducrot Bruno)

Kobject fix (Greg KH)

3 iASL Compiler/Disassembler:

Fixed a problem in the generation of the C source code files (AML is emitted in C source statements for BIOS inclusion) where the

Ascii dump that appears within a C comment at the end of each line
could cause a compile time error if the AML sequence happens to
have an open comment or close comment sequence embedded.

---
24 April 2003. Summary of changes for version 20030424.

1) ACPI CA Core Subsystem:

Support for big-endian systems has been implemented. Most of the
support has been invisibly added behind big-endian versions of the
ACPI_MOVE_* macros.

Fixed a problem in AcpiHwDisableGpeBlock() and
AcpiHwClearGpeBlock() where an incorrect offset was passed to the
low level hardware write routine. The offset parameter was
actually eliminated from the low level read/write routines because
they had become obsolete.

Fixed a problem where a handler object was deleted twice during
the removal of a fixed event handler.

2) Linux:

A fix for SMP systems with link devices was contributed by

Compaq's Dan Zink.

(2.5) Return whether we handled the interrupt in our IRQ handler.
(Linux ISRs no longer return void, so we can propagate the handler
return value from the ACPI CA core back to the OS.)

3) Documentation:

The ACPI CA Programmer Reference has been updated to reflect new
interfaces and changes to existing interfaces.

---
28 March 2003. Summary of changes for version 20030328.

1) ACPI CA Core Subsystem:

The GPE Block Device support has been completed. New interfaces
are AcpiInstallGpeBlock and AcpiRemoveGpeBlock. The Event
interfaces (enable, disable, clear, getstatus) have been split
into separate interfaces for Fixed Events and General Purpose
Events (GPEs) in order to support GPE Block Devices properly.

Fixed a problem where the error message "Failed to acquire
semaphore" would appear during operations on the embedded
controller (EC).

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

  Previous Release
    Non-Debug Version: 72.3K Code,   9.5K Data,   81.8K Total
    Debug Version: 154.0K Code, 63.4K Data, 217.4K Total
  Current Release:
    Non-Debug Version: 73.7K Code,   9.5K Data,   83.2K Total
    Debug Version: 156.1K Code, 63.6K Data, 219.7K Total

---
28 February 2003. Summary of changes for version 20030228.

1) ACPI CA Core Subsystem:

The GPE handling and dispatch code has been completely overhauled
in preparation for support of GPE Block Devices (ID ACPI0006).
This affects internal data structures and code only; there should
be no differences visible externally. One new file has been
added, evgpeblk.c

The FADT fields GPE0_BLK_LEN and GPE1_BLK_LEN are now the only
fields that are used to determine the GPE block lengths. The
REGISTER_BIT_WIDTH field of the X_GPEx_BLK extended address
structures are ignored. This is per the ACPI specification but it
isn't very clear. The full 256 Block 0/1 GPEs are now supported
(the use of REGISTER_BIT_WIDTH limited the number of GPEs to 128).

In the SCI interrupt handler, removed the read of the PM1_CONTROL
register to look at the SCI_EN bit. On some machines, this read
causes an SMI event and greatly slows down SCI events. (This may
in fact be the cause of slow battery status response on some
systems.)

Fixed a problem where a store of a NULL string to a package object
could cause the premature deletion of the object. This was seen
during execution of the battery _BIF method on some systems,
resulting in no battery data being returned.

Added AcpiWalkResources interface to simplify parsing of resource
lists.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

  Previous Release
    Non-Debug Version: 72.0K Code,   9.5K Data,   81.5K Total
    Debug Version: 153.0K Code, 62.9K Data, 215.9K Total
  Current Release:
    Non-Debug Version: 72.3K Code,   9.5K Data,   81.8K Total
    Debug Version: 154.0K Code, 63.4K Data, 217.4K Total

2) Linux

S3 fixes (Ole Rohne)

Update ACPI PHP driver with to use new acpi_walk_resource API
(Bjorn Helgaas)

Add S4BIOS support (Pavel Machek)

Map in entire table before performing checksum (John Stultz)

Expand the mem= cmdline to allow the specification of reserved and
ACPI DATA blocks (Pavel Machek)

Never use ACPI on VISWS

Fix derive_pci_id (Ducrot Bruno, Alvaro Lopez)

Revert a change that allowed P_BLK lengths to be 4 or 5. This is
causing us to think that some systems support C2 when they really
don't.

Do not count processor objects for non-present CPUs (Thanks to
Dominik Brodowski)

3) iASL Compiler:

Fixed a problem where ASL include files could not be found and
opened.

Added support for the _PDC reserved name.

---
22 January 2003. Summary of changes for version 20030122.

1) ACPI CA Core Subsystem:

Added a check for constructs of the form: Store (Local0, Local0)
where Local0 is not initialized. Apparently, some BIOS
programmers believe that this is a NOOP. Since this store doesn't
do anything anyway, the new prototype behavior will ignore this
error. This is a case where we can relax the strict checking in
the interpreter in the name of compatibility.

2) Linux

The AcpiSrc Source Conversion Utility has been released with the
Linux package for the first time. This is the utility that is
used to convert the ACPI CA base source code to the Linux version.

(Both) Handle P_BLK lengths shorter than 6 more gracefully

(Both) Move more headers to include/acpi, and delete an unused
header.

(Both) Move drivers/acpi/include directory to include/acpi

(Both) Boot functions don't use cmdline, so don't pass it around

(Both) Remove include of unused header (Adrian Bunk)

(Both) acpiphp.h includes both linux/acpi.h and acpi_bus.h. Since
the
former now also includes the latter, acpiphp.h only needs the one,
now.

(2.5) Make it possible to select method of bios restoring after S3
resume. [=> no more ugly ifdefs] (Pavel Machek)

(2.5) Make proc write interfaces work (Pavel Machek)

(2.5) Properly init/clean up in cpufreq/acpi (Dominik Brodowski)

(2.5) Break out ACPI Perf code into its own module, under cpufreq
(Dominik Brodowski)

(2.4) S4BIOS support (Ducrot Bruno)

(2.4) Fix acpiphp_glue.c for latest ACPI struct changes (Sergio
Visinoni)

3) iASL Compiler:

Added support to disassemble SSDT and PSDTs.

Implemented support to obtain SSDTs from the Windows registry if
available.

---
09 January 2003. Summary of changes for version 20030109.

1) ACPI CA Core Subsystem:

Changed the behavior of the internal Buffer-to-String conversion
function. The current ACPI specification states that the contents
of the buffer are "converted to a string of two-character
hexadecimal numbers, each separated by a space". Unfortunately,
this definition is not backwards compatible with existing ACPI 1.0
implementations (although the behavior was not defined in the ACPI
1.0 specification). The new behavior simply copies data from the
buffer to the string until a null character is found or the end of
the buffer is reached. The new String object is always null
terminated. This problem was seen during the generation of _BIF
battery data where incorrect strings were returned for battery
type, etc. This will also require an errata to the ACPI
specification.

Renamed all instances of NATIVE_UINT and NATIVE_INT to
ACPI_NATIVE_UINT and ACPI_NATIVE_INT, respectively.

Copyright in all module headers (both Linux and non-Linux) has be
updated to 2003.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

  Previous Release
    Non-Debug Version: 72.0K Code,   9.5K Data,   81.5K Total
    Debug Version: 153.0K Code, 62.9K Data, 215.9K Total
  Current Release:
    Non-Debug Version: 72.0K Code,   9.5K Data,   81.5K Total
    Debug Version: 153.0K Code, 62.9K Data, 215.9K Total

2) Linux

Fixed an oops on module insertion/removal (Matthew Tippett)

11309 (2.4) Fix to handle dynamic size of mp_irqs (Joerg Prante)

11311 (2.5) Replace pr_debug (Randy Dunlap)

11313 (2.5) Remove usage of CPUFREQ_ALL_CPUS (Dominik Brodowski)

11315 (Both) Eliminate spawning of thread from timer callback, in favor
11316 of schedule_work()

11318 (Both) Show Lid status in /proc (Zdenek OGAR Skalak)

11320 (Both) Added define for Fixed Function HW region (Matthew Wilcox)

11322 (Both) Add missing statics to button.c (Pavel Machek)

11324 Several changes have been made to the source code translation
11325 utility that generates the Linux Code in order to make the code
11326 more "Linux-like":

11328 All typedefs on structs and unions have been removed in keeping
11329 with the Linux coding style.

11331 Removed the non-Linux SourceSafe module revision number from each
11332 module header.

11334 Completed major overhaul of symbols to be lowercased for linux.
11335 Doubled the number of symbols that are lowercased.

11337 Fixed a problem where identifiers within procedure headers and
11338 within quotes were not fully lower cased (they were left with a
11339 starting capital.)

11341 Some C macros whose only purpose is to allow the generation of 16-
11342 bit code are now completely removed in the Linux code, increasing
11343 readability and maintainability.

11345 --

11347 12 December 2002. Summary of changes for version 20021212.

11350 1) ACPI CA Core Subsystem:

11352 Fixed a problem where the creation of a zero-length AML Buffer
11353 would cause a fault.

11355 Fixed a problem where a Buffer object that pointed to a static AML
11356 buffer (in an ACPI table) could inadvertently be deleted, causing
11357 memory corruption.

11359 Fixed a problem where a user buffer (passed in to the external
11360 ACPI CA interfaces) could be overwritten if the buffer was too
11361 small to complete the operation, causing memory corruption.

11363 Fixed a problem in the Buffer-to-String conversion code where a
11364 string of length one was always returned, regardless of the size
11365 of the input Buffer object.

11367 Removed the NATIVE_CHAR data type across the entire source due to
11368 lack of need and lack of consistent use.

11370 Code and Data Size: Current core subsystem library sizes are shown
11371 below. These are the code and data sizes for the acpica.lib
11372 produced by the Microsoft Visual C++ 6.0 compiler, and these
11373 values do not include any ACPI driver or OSPM code. The debug
11374 version of the code includes the debug output trace mechanism and

11375 has a much larger code and data size. Note that these values will
11376 vary depending on the efficiency of the compiler and the compiler
11377 options used during generation.

11379   Previous Release
11380   Non-Debug Version: 72.1K Code,   9.5K Data,   81.6K Total
11381   Debug Version: 152.7K Code, 62.7K Data, 215.4K Total
11382   Current Release:
11383   Non-Debug Version: 72.0K Code,   9.5K Data,   81.5K Total
11384   Debug Version: 153.0K Code, 62.9K Data, 215.9K Total

11387 --
11388 05 December 2002. Summary of changes for version 20021205.

11390 1) ACPI CA Core Subsystem:

11392 Fixed a problem where a store to a String or Buffer object could
11393 cause corruption of the DSDT if the object type being stored was
11394 the same as the target object type and the length of the object
11395 being stored was equal to or smaller than the original (existing)
11396 target object. This was seen to cause corruption of battery _BIF
11397 buffers if the _BIF method modified the buffer on the fly.

11399 Fixed a problem where an internal error was generated if a control
11400 method invocation was used in an OperationRegion, Buffer, or
11401 Package declaration. This was caused by the deferred parsing of
11402 the control method and thus the deferred creation of the internal
11403 method object. The solution to this problem was to create the
11404 internal method object at the moment the method is encountered in
11405 the first pass - so that subsequent references to the method will
11406 able to obtain the required parameter count and thus properly
11407 parse the method invocation. This problem presented itself as an
11408 AE_AML_INTERNAL during the pass 1 parse phase during table load.

11410 Fixed a problem where the internal String object copy routine did
11411 not always allocate sufficient memory for the target String object
11412 and caused memory corruption. This problem was seen to cause
11413 "Allocation already present in list!" errors as memory allocation
11414 became corrupted.

11416 Implemented a new function for the evaluation of namespace objects
11417 that allows the specification of the allowable return object
11418 types. This simplifies a lot of code that checks for a return
11419 object of one or more specific objects returned from the
11420 evaluation (such as _STA, etc.) This may become and external
11421 function if it would be useful to ACPI-related drivers.

11423 Completed another round of prefixing #defines with "ACPI_" for
11424 clarity.

11426 Completed additional code restructuring to allow more modular
11427 linking for iASL compiler and AcpiExec. Several files were split
11428 creating new files. New files: nsparse.c dsinit.c evgpe.c

11430 Implemented an abort mechanism to terminate an executing control
11431 method via the AML debugger. This feature is useful for debugging
11432 control methods that depend (wait) for specific hardware
11433 responses.

11435 Code and Data Size: Current core subsystem library sizes are shown
11436 below. These are the code and data sizes for the acpica.lib
11437 produced by the Microsoft Visual C++ 6.0 compiler, and these
11438 values do not include any ACPI driver or OSPM code. The debug
11439 version of the code includes the debug output trace mechanism and
11440 has a much larger code and data size. Note that these values will

vary depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  71.4K Code,   9.0K Data,   80.4K Total
    Debug Version:     152.9K Code,  63.3K Data,  216.2K Total
  Current Release:
    Non-Debug Version:  72.1K Code,   9.5K Data,   81.6K Total
    Debug Version:     152.7K Code,  62.7K Data,  215.4K Total
```

2) iASL Compiler/Disassembler

Fixed a compiler code generation problem for "Interrupt" Resource
Descriptors. If specified in the ASL, the optional "Resource
Source Index" and "Resource Source" fields were not inserted into
the correct location within the AML resource descriptor, creating
an invalid descriptor.

Fixed a disassembler problem for "Interrupt" resource descriptors.
The optional "Resource Source Index" and "Resource Source" fields
were ignored.

---

22 November 2002. Summary of changes for version 20021122.

1) ACPI CA Core Subsystem:

Fixed a reported problem where an object stored to a Method Local
or Arg was not copied to a new object during the store - the
object pointer was simply copied to the Local/Arg. This caused
all subsequent operations on the Local/Arg to also affect the
original source of the store operation.

Fixed a problem where a store operation to a Method Local or Arg
was not completed properly if the Local/Arg contained a reference
(from RefOf) to a named field. The general-purpose store-to-
namespace-node code is now used so that this case is handled
automatically.

Fixed a problem where the internal object copy routine would cause
a protection fault if the object being copied was a Package and
contained either 1) a NULL package element or 2) a nested sub-
package.

Fixed a problem with the GPE initialization that resulted from an
ambiguity in the ACPI specification. One section of the
specification states that both the address and length of the GPE
block must be zero if the block is not supported. Another section
implies that only the address need be zero if the block is not
supported. The code has been changed so that both the address and
the length must be non-zero to indicate a valid GPE block (i.e.,
if either the address or the length is zero, the GPE block is
invalid.)

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.


```
  Previous Release
    Non-Debug Version:  71.3K Code,   9.0K Data,   80.3K Total
    Debug Version:     152.7K Code,  63.2K Data,  215.5K Total
  Current Release:
    Non-Debug Version:  71.4K Code,   9.0K Data,   80.4K Total
    Debug Version:     152.9K Code,  63.3K Data,  216.2K Total
```

2) Linux

Cleaned up EC driver. Exported an external EC read/write
interface. By going through this, other drivers (most notably
sonypi) will be able to serialize access to the EC.

3) iASL Compiler/Disassembler

Implemented support to optionally generate include files for both
ASM and C (the -i switch). This simplifies BIOS development by
automatically creating include files that contain external
declarations for the symbols that are created within the

(optionally generated) ASM and C AML source files.

---

15 November 2002. Summary of changes for version 20021115.

1) ACPI CA Core Subsystem:

Fixed a memory leak problem where an error during resolution of

method arguments during a method invocation from another method
failed to cleanup properly by deleting all successfully resolved
argument objects.

Fixed a problem where the target of the Index() operator was not
correctly constructed if the source object was a package. This
problem has not been detected because the use of a target operand
with Index() is very rare.

Fixed a problem with the Index() operator where an attempt was
made to delete the operand objects twice.

Fixed a problem where an attempt was made to delete an operand
twice during execution of the CondRefOf() operator if the target
did not exist.

Implemented the first of perhaps several internal create object
functions that create and initialize a specific object type. This
consolidates duplicated code wherever the object is created, thus
shrinking the size of the subsystem.

Implemented improved debug/error messages for errors that occur
during nested method invocations. All executing method pathnames
are displayed (with the error) as the call stack is unwound - thus
simplifying debug.

Fixed a problem introduced in the 10/02 release that caused
premature deletion of a buffer object if a buffer was used as an
ASL operand where an integer operand is required (Thus causing an
implicit object conversion from Buffer to Integer.) The change in
the 10/02 release was attempting to fix a memory leak (albeit
incorrectly.)

Code and Data Size: Current core subsystem library sizes are shown

below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  71.9K Code,   9.1K Data,   81.0K Total
    Debug Version:     153.1K Code,  63.3K Data,  216.4K Total
  Current Release:
    Non-Debug Version:  71.3K Code,   9.0K Data,   80.3K Total
    Debug Version:     152.7K Code,  63.2K Data,  215.5K Total
```

2) Linux

Changed the implementation of the ACPI semaphores to use down()
instead of down_interruptable(). It is important that the
execution of ACPI control methods not be interrupted by signals.
Methods must run to completion, or the system may be left in an
unknown/unstable state.

Fixed a compilation error when CONFIG_SOFTWARE_SUSPEND is not set.
(Shawn Starr)

3) iASL Compiler/Disassembler

Changed the default location of output files. All output files
are now placed in the current directory by default instead of in
the directory of the source file. This change may affect some
existing makefiles, but it brings the behavior of the compiler in
line with other similar tools. The location of the output files
can be overridden with the -p command line switch.

---
11 November 2002. Summary of changes for version 20021111.

0) ACPI Specification 2.0B is released and is now available at:
http://www.acpi.info/index.html

1) ACPI CA Core Subsystem:

Implemented support for the ACPI 2.0 SMBus Operation Regions.
This includes the early detection and handoff of the request to
the SMBus region handler (avoiding all of the complex field
support code), and support for the bidirectional return packet
from an SMBus write operation. This paves the way for the
development of SMBus drivers in each host operating system.

Fixed a problem where the semaphore WAIT_FOREVER constant was
defined as 32 bits, but must be 16 bits according to the ACPI
specification. This had the side effect of causing ASL
Mutex/Event timeouts even though the ASL code requested a wait
forever. Changed all internal references to the ACPI timeout
parameter to 16 bits to prevent future problems. Changed the name
of WAIT_FOREVER to ACPI_WAIT_FOREVER.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib

produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  71.4K Code,   9.0K Data,   80.4K Total
    Debug Version:     152.3K Code,  63.0K Data,  215.3K Total
  Current Release:
    Non-Debug Version:  71.9K Code,   9.1K Data,   81.0K Total
    Debug Version:     153.1K Code,  63.3K Data,  216.4K Total
```

2) Linux

Module loading/unloading fixes (John Cagle)

3) iASL Compiler/Disassembler

Added support for the SMBBlockProcessCall keyword (ACPI 2.0)

Implemented support for the disassembly of all SMBus protocol
keywords (SMBQuick, SMBWord, etc.)

---
01 November 2002. Summary of changes for version 20021101.

1) ACPI CA Core Subsystem:

Fixed a problem where platforms that have a GPE1 block but no GPE0
block were not handled correctly. This resulted in a "GPE
overlap" error message. GPE0 is no longer required.

Removed code added in the previous release that inserted nodes
into the namespace in alphabetical order. This caused some side-
effects on various machines. The root cause of the problem is
still under investigation since in theory, the internal ordering
of the namespace nodes should not matter.

Enhanced error reporting for the case where a named object is not
found during control method execution. The full ACPI namepath
(name reference) of the object that was not found is displayed in
this case.

Note: as a result of the overhaul of the namespace object types in
the previous release, the namespace nodes for the predefined
scopes (_TZ, _PR, etc.) are now of the type ACPI_TYPE_LOCAL_SCOPE
instead of ACPI_TYPE_ANY. This simplifies the namespace
management code but may affect code that walks the namespace tree
looking for specific object types.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a much larger code and data size. Note that these values will
vary depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
```

```
11705     Non-Debug Version:  70.7K Code,   8.6K Data,   79.3K Total
11706     Debug Version:     151.7K Code,  62.4K Data,  214.1K Total
11707   Current Release:
11708     Non-Debug Version:  71.4K Code,   9.0K Data,   80.4K Total
11709     Debug Version:     152.3K Code,  63.0K Data,  215.3K Total


11712 2) Linux

11714 Fixed a problem introduced in the previous release where the
11715 Processor and Thermal objects were not recognized and installed in
11716 /proc.  This was related to the scope type change described above.


11719 3) iASL Compiler/Disassembler

11721 Implemented the -g option to get all of the required ACPI tables
11722 from the registry and save them to files (Windows version of the
11723 compiler only.)  The required tables are the FADT, FACS, and DSDT.

11725 Added ACPI table checksum validation during table disassembly in
11726 order to catch corrupted tables.


11729 ----------------------------------------
11730 22 October 2002.  Summary of changes for version 20021022.

11732 1) ACPI CA Core Subsystem:

11734 Implemented a restriction on the Scope operator that the target
11735 must already exist in the namespace at the time the operator is
11736 encountered (during table load or method execution).  In other
11737 words, forward references are not allowed and Scope() cannot
11738 create a new object. This changes the previous behavior where the
11739 interpreter would create the name if not found.  This new behavior
11740 correctly enables the search-to-root algorithm during namespace
11741 lookup of the target name.  Because of this upsearch, this fixes
11742 the known Compaq _SB_.OKEC problem and makes both the AML
11743 interpreter and iASL compiler compatible with other ACPI
11744 implementations.

11746 Completed a major overhaul of the internal ACPI object types for
11747 the ACPI Namespace and the associated operand objects.  Many of
11748 these types had become obsolete with the introduction of the two-
11749 pass namespace load.  This cleanup simplifies the code and makes
11750 the entire namespace load mechanism much clearer and easier to
11751 understand.

11753 Improved debug output for tracking scope opening/closing to help
11754 diagnose scoping issues.  The old scope name as well as the new
11755 scope name are displayed.  Also improved error messages for
11756 problems with ASL Mutex objects and error messages for GPE
11757 problems.

11759 Cleaned up the namespace dump code, removed obsolete code.

11761 All string output (for all namespace/object dumps) now uses the
11762 common ACPI string output procedure which handles escapes properly
11763 and does not emit non-printable characters.

11765 Fixed some issues with constants in the 64-bit version of the
11766 local C library (utclib.c)


11769 2) Linux
```



```
11771 EC Driver:  No longer attempts to acquire the Global Lock at
11772 interrupt level.


11775 3) iASL Compiler/Disassembler

11777 Implemented ACPI 2.0B grammar change that disallows all Type 1 and
11778 2 opcodes outside of a control method.  This means that the
11779 "executable" operators (versus the "namespace" operators) cannot
11780 be used at the table level; they can only be used within a control
11781 method.

11783 Implemented the restriction on the Scope() operator where the
11784 target must already exist in the namespace at the time the
11785 operator is encountered (during ASL compilation). In other words,
11786 forward references are not allowed and Scope() cannot create a new
11787 object.  This makes the iASL compiler compatible with other ACPI
11788 implementations and makes the Scope() implementation adhere to the
11789 ACPI specification.

11791 Fixed a problem where namepath optimization for the Alias operator
11792 was optimizing the wrong path (of the two namepaths.)  This caused
11793 a "Missing alias link" error message.

11795 Fixed a problem where an "unknown reserved name" warning could be
11796 incorrectly generated for names like "_SB" when the trailing
11797 underscore is not used in the original ASL.

11799 Fixed a problem where the reserved name check did not handle
11800 NamePaths with multiple NameSegs correctly.  The first nameseg of
11801 the NamePath was examined instead of the last NameSeg.


11804 ----------------------------------------

11806 02 October 2002.  Summary of changes for this release.


11809 1) ACPI CA Core Subsystem version 20021002:

11811 Fixed a problem where a store/copy of a string to an existing
11812 string did not always set the string length properly in the String
11813 object.

11815 Fixed a reported problem with the ToString operator where the
11816 behavior was identical to the ToHexString operator instead of just
11817 simply converting a raw buffer to a string data type.

11819 Fixed a problem where CopyObject and the other "explicit"
11820 conversion operators were not updating the internal namespace node
11821 type as part of the store operation.

11823 Fixed a memory leak during implicit source operand conversion
11824 where the original object was not deleted if it was converted to a
11825 new object of a different type.

11827 Enhanced error messages for all problems associated with namespace
11828 lookups.  Common procedure generates and prints the lookup name as
11829 well as the formatted status.

11831 Completed implementation of a new design for the Alias support
11832 within the namespace.  The existing design did not handle the case
11833 where a new object was assigned to one of the two names due to the
11834 use of an explicit conversion operator, resulting in the two names
11835 pointing to two different objects.  The new design simply points
11836 the Alias name to the original name node - not to the object.
```

This results in a level of indirection that must be handled in the name resolution mechanism.

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release
    Non-Debug Version:  69.6K Code,   8.3K Data,   77.9K Total
    Debug Version:     150.0K Code,  61.7K Data,  211.7K Total
  Current Release:
    Non-Debug Version:  70.7K Code,   8.6K Data,   79.3K Total
    Debug Version:     151.7K Code,  62.4K Data,  214.1K Total
```

2) Linux

Initialize thermal driver’s timer before it is used. (Knut Neumann)

Allow handling negative celsius values. (Kochi Takayoshi)

Fix thermal management and make trip points. R/W (Pavel Machek)

Fix /proc/acpi/sleep. (P. Christeas)

IA64 fixes. (David Mosberger)

Fix reversed logic in blacklist code. (Sergio Monteiro Basto)

Replace ACPI_DEBUG define with ACPI_DEBUG_OUTPUT. (Dominik Brodowski)

3) iASL Compiler/Disassembler

Clarified some warning/error messages.

---

18 September 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem version 20020918:

Fixed a reported problem with reference chaining (via the Index() and RefOf() operators) in the ObjectType() and SizeOf() operators. The definition of these operators includes the dereferencing of all chained references to return information on the base object.

Fixed a problem with stores to indexed package elements - the existing code would not complete the store if an "implicit conversion" was not performed. In other words, if the existing object (package element) was to be replaced completely, the code didn’t handle this case.

Relaxed typechecking on the ASL "Scope" operator to allow the target name to refer to an object of type Integer, String, or Buffer, in addition to the scoping object types (Device, predefined Scopes, Processor, PowerResource, and ThermalZone.) This allows existing AML code that has workarounds for a bug in

Windows to function properly. A warning is issued, however. This affects both the AML interpreter and the iASL compiler. Below is an example of this type of ASL code:

```
      Name(DEB,0x00)
      Scope(DEB)
      {
```

Fixed some reported problems with 64-bit integer support in the local implementation of C library functions (clib.c)

2) Linux

Use ACPI fix map region instead of IOAPIC region, since it is undefined in non-SMP.

Ensure that the SCI has the proper polarity and trigger, even on systems that do not have an interrupt override entry in the MADT.

2.5 big driver reorganization (Pat Mochel)

Use early table mapping code from acpitable.c (Andi Kleen)

New blacklist entries (Andi Kleen)

Blacklist improvements. Split blacklist code out into a separate file. Move checking the blacklist to very early. Previously, we would use ACPI tables, and then halfway through init, check the blacklist -- too late. Now, it’s early enough to completely fall-back to non-ACPI.

3) iASL Compiler/Disassembler version 20020918:

Fixed a problem where the typechecking code didn’t know that an alias could point to a method. In other words, aliases were not being dereferenced during typechecking.

---

29 August 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020829:

If the target of a Scope() operator already exists, it must be an object type that actually opens a scope -- such as a Device, Method, Scope, etc. This is a fatal runtime error. Similar error check has been added to the iASL compiler also.

Tightened up the namespace load to disallow multiple names in the same scope. This previously was allowed if both objects were of the same type. (i.e., a lookup was the same as entering a new name).

2) Linux

Ensure that the ACPI interrupt has the proper trigger and polarity.

local_irq_disable is extraneous. (Matthew Wilcox)

Make "acpi=off" actually do what it says, and not use the ACPI interpreter *or* the tables.

Added arch-neutral support for parsing SLIT and SRAT tables (Kochi Takayoshi)

3) iASL Compiler/Disassembler Version 20020829:

Implemented namepath optimization for name declarations. For example, a declaration like "Method (_SB_.ABCD)" would get optimized to "Method (ABCD)" if the declaration is within the _SB_ scope. This optimization is in addition to the named reference path optimization first released in the previous version. This would seem to complete all possible optimizations for namepaths within the ASL/AML.

If the target of a Scope() operator already exists, it must be an object type that actually opens a scope -- such as a Device, Method, Scope, etc.

Implemented a check and warning for unreachable code in the same block below a Return() statement.

Fixed a problem where the listing file was not generated if the compiler aborted if the maximum error count was exceeded (200).

Fixed a problem where the typechecking of method return values was broken. This includes the check for a return value when the method is invoked as a TermArg (a return value is expected.)

Fixed a reported problem where EOF conditions during a quoted string or comment caused a fault.

---
15 August 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020815:

Fixed a reported problem where a Store to a method argument that contains a reference did not perform the indirect store correctly. This problem was created during the conversion to the new reference object model - the indirect store to a method argument code was not updated to reflect the new model.

Reworked the ACPI mode change code to better conform to ACPI 2.0, handle corner cases, and improve code legibility (Kochi Takayoshi)

Fixed a problem with the pathname parsing for the carat (^) prefix. The heavy use of the carat operator by the new namepath optimization in the iASL compiler uncovered a problem with the AML interpreter handling of this prefix. In the case where one or more carats precede a single nameseg, the nameseg was treated as standalone and the search rule (to root) was inadvertently applied. This could cause both the iASL compiler and the interpreter to find the wrong object or to miss the error that should occur if the object does not exist at that exact pathname.

Found and fixed the problem where the HP Pavilion DSDT would not load. This was a relatively minor tweak to the table loading code (a problem caused by the unexpected encounter with a method invocation not within a control method), but it does not solve the overall issue of the execution of AML code at the table level. This investigation is still ongoing.

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these

values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release
    Non-Debug Version:  69.1K Code,   8.2K Data,   77.3K Total
    Debug Version:     149.4K Code,  61.6K Data,  211.0K Total
  Current Release:
    Non-Debug Version:  69.6K Code,   8.3K Data,   77.9K Total
    Debug Version:     150.0K Code,  61.7K Data,  211.7K Total
```

2) Linux

Remove redundant slab.h include (Brad Hards)

Fix several bugs in thermal.c (Herbert Nachtnebel)

Make CONFIG_ACPI_BOOT work properly (Pavel Machek)

Change acpi_system_suspend to use updated irq functions (Pavel Machek)

Export acpi_get_firmware_table (Matthew Wilcox)

Use proper root proc entry for ACPI (Kochi Takayoshi)

Fix early-boot table parsing (Bjorn Helgaas)

3) iASL Compiler/Disassembler

Reworked the compiler options to make them more consistent and to use two-letter options where appropriate. We were running out of sensible letters.   This may break some makefiles, so check the current options list by invoking the compiler with no parameters.

Completed the design and implementation of the ASL namepath optimization option for the compiler. This option optimizes all references to named objects to the shortest possible path. The first attempt tries to utilize a single nameseg (4 characters) and the "search-to-root" algorithm used by the interpreter. If that cannot be used (because either the name is not in the search path or there is a conflict with another object with the same name), the pathname is optimized using the carat prefix (usually a shorter string than specifying the entire path from the root.)

Implemented support to obtain the DSDT from the Windows registry (when the disassembly option is specified with no input file). Added this code as the implementation for AcpiOsTableOverride in the Windows OSL. Migrated the 16-bit code (used in the AcpiDump utility) to scan memory for the DSDT to the AcpiOsTableOverride function in the DOS OSL to make the disassembler truly OS independent.

Implemented a new option to disassemble and compile in one step. When used without an input filename, this option will grab the DSDT from the local machine, disassemble it, and compile it in one step.

Added a warning message for invalid escapes (a backslash followed by any character other than the allowable escapes). This catches the quoted string error "_SB_" (which should be "_SB_").

Also, there are numerous instances in the ACPI specification where this error occurs.

Added a compiler option to disable all optimizations. This is basically the "compatibility mode" because by using this option, the AML code will come out exactly the same as other ASL compilers.

Added error messages for incorrectly ordered dependent resource functions. This includes: missing EndDependentFn macro at end of dependent resource list, nested dependent function macros (both start and end), and missing StartDependentFn macro. These are common errors that should be caught at compile time.

Implemented _OSI support for the disassembler and compiler. _OSI must be included in the namespace for proper disassembly (because the disassembler must know the number of arguments.)

Added an "optimization" message type that is optional (off by default). This message is used for all optimizations - including constant folding, integer optimization, and namepath optimization.

---
25 July 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020725:

The AML Disassembler has been enhanced to produce compilable ASL code and has been integrated into the iASL compiler (see below) as well as the single-step disassembly for the AML debugger and the disassembler for the AcpiDump utility. All ACPI 2.0A opcodes, resource templates and macros are fully supported. The disassembler has been tested on over 30 different AML files, producing identical AML when the resulting disassembled ASL file is recompiled with the same ASL compiler.

Modified the Resource Manager to allow zero interrupts and zero dma channels during the GetCurrentResources call. This was causing problems on some platforms.

Added the AcpiOsRedirectOutput interface to the OSL to simplify output redirection for the AcpiOsPrintf and AcpiOsVprintf interfaces.

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release
    Non-Debug Version:  68.7K Code,   7.4K Data,   76.1K Total
    Debug Version:     142.9K Code,  58.7K Data,  201.6K Total
  Current Release:
    Non-Debug Version:  69.1K Code,   8.2K Data,   77.3K Total
    Debug Version:     149.4K Code,  61.6K Data,  211.0K Total
```

2) Linux

Fixed a panic in the EC driver (Dominik Brodowski)

Implemented checksum of the R/XSDT itself during Linux table scan (Richard Schaal)

3) iASL compiler

The AML disassembler is integrated into the compiler. The "-d" option invokes the disassembler to completely disassemble an input AML file, producing as output a text ASL file with the extension ".dsl" (to avoid name collisions with existing .asl source files.) A future enhancement will allow the disassembler to obtain the BIOS DSDT from the registry under Windows.

Fixed a problem with the VendorShort and VendorLong resource descriptors where an invalid AML sequence was created.

Implemented a fix for BufferData term in the ASL parser. It was inadvertently defined twice, allowing invalid syntax to pass and causing reduction conflicts.

Fixed a problem where the Ones opcode could get converted to a value of zero if "Ones" was used where a byte, word or dword value was expected. The 64-bit value is now truncated to the correct size with the correct value.

---
02 July 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020702:

The Table Manager code has been restructured to add several new features. Tables that are not required by the core subsystem (other than the FADT, DSDT, FACS, PSDTs, etc.) are no longer validated in any way and are returned from AcpiGetFirmwareTable if requested. The AcpiOsTableOverride interface is now called for each table that is loaded by the subsystem in order to allow the host to override any table it chooses. Previously, only the DSDT could be overridden. Added one new files, tbrsdt.c and tbgetall.c.

Fixed a problem with the conversion of internal package objects to external objects (when a package is returned from a control method.) The return buffer length was set to zero instead of the proper length of the package object.

Fixed a reported problem with the use of the RefOf and DeRefOf operators when passing reference arguments to control methods. A new type of Reference object is used internally for references produced by the RefOf operator.

Added additional error messages in the Resource Manager to explain AE_BAD_DATA errors when they occur during resource parsing.

Split the AcpiEnableSubsystem into two primitives to enable a finer granularity initialization sequence. These two calls should be called in this order: AcpiEnableSubsystem (flags), AcpiInitializeObjects (flags). The flags parameter remains the same.

2) Linux

Updated the ACPI utilities module to understand the new style of

fully resolved package objects that are now returned from the core
subsystem. This eliminates errors of the form:

```
    ACPI: PCI Interrupt Routing Table [\_SB_.PCI0.PPB_._PRT]
    acpi_utils-0430 [145] acpi_evaluate_reference:
        Invalid element in package (not a device reference)
```

The method evaluation utility uses the new buffer allocation
scheme instead of calling AcpiEvaluate Object twice.

Added support for ECDT. This allows the use of the Embedded

Controller before the namespace has been fully initialized, which
is necessary for ACPI 2.0 support, and for some laptops to
initialize properly. (Laptops using ECDT are still rare, so only
limited testing was performed of the added functionality.)

Fixed memory leaks in the EC driver.

Eliminated a brittle code structure in acpi_bus_init().

Eliminated the acpi_evaluate() helper function in utils.c. It is
no longer needed since acpi_evaluate_object can optionally
allocate memory for the return object.

Implemented fix for keyboard hang when getting battery readings on
some systems (Stephen White)

PCI IRQ routing update (Dominik Brodowski)

Fix an ifdef to allow compilation on UP with LAPIC but no IOAPIC
support

---

11 June 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020611:

Fixed a reported problem where constants such as Zero and One
appearing within _PRT packages were not handled correctly within
the resource manager code. Originally reported against the ASL
compiler because the code generator now optimizes integers to
their minimal AML representation (i.e. AML constants if possible.)
The _PRT code now handles all AML constant opcodes correctly
(Zero, One, Ones, Revision).

Fixed a problem with the Concatenate operator in the AML
interpreter where a buffer result object was incorrectly marked as
not fully evaluated, causing a run-time error of AE_AML_INTERNAL.

All package sub-objects are now fully resolved before they are
returned from the external ACPI interfaces. This means that name
strings are resolved to object handles, and constant operators
(Zero, One, Ones, Revision) are resolved to Integers.

Implemented immediate resolution of the AML Constant opcodes
(Zero, One, Ones, Revision) to Integer objects upon detection
within the AML stream. This has simplified and reduced the
generated code size of the subsystem by eliminating about 10
switch statements for these constants (which previously were
contained in Reference objects.) The complicating issues are that
the Zero opcode is used as a "placeholder" for unspecified
optional target operands and stores to constants are defined to be
no-ops.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  69.3K Code,   7.4K Data,   76.7K Total
    Debug Version:     143.8K Code,  58.8K Data,  202.6K Total
  Current Release:
    Non-Debug Version:  68.7K Code,   7.4K Data,   76.1K Total
    Debug Version:     142.9K Code,  58.7K Data,  201.6K Total
```

2) Linux

Added preliminary support for obtaining _TRA data for PCI root
bridges (Bjorn Helgaas).

3) iASL Compiler Version X2046:

Fixed a problem where the "_DDN" reserved name was defined to be a
control method with one argument. There are no arguments, and
_DDN does not have to be a control method.

Fixed a problem with the Linux version of the compiler where the
source lines printed with error messages were the wrong lines.
This turned out to be the "LF versus CR/LF" difference between
Windows and Unix. This appears to be the longstanding issue
concerning listing output and error messages.

Fixed a problem with the Linux version of compiler where opcode
names within error messages were wrong. This was caused by a
slight difference in the output of the Flex tool on Linux versus
Windows.

Fixed a problem with the Linux compiler where the hex output files
contained some garbage data caused by an internal buffer overrun.

---

17 May 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020517:

Implemented a workaround to an BIOS bug discovered on the HP
OmniBook where the FADT revision number and the table size are
inconsistent (ACPI 2.0 revision vs. ACPI 1.0 table size). The new
behavior is to fallback to using only the ACPI 1.0 fields of the
FADT if the table is too small to be a ACPI 2.0 table as claimed
by the revision number. Although this is a BIOS bug, this is a
case where the workaround is simple enough and with no side
effects, so it seemed prudent to add it. A warning message is
issued, however.

Implemented minimum size checks for the fixed-length ACPI tables -
- the FADT and FACS, as well as consistency checks between the
revision number and the table size.

Fixed a reported problem in the table override support where the

new table pointer was incorrectly treated as a physical address
instead of a logical address.

Eliminated the use of the AE_AML_ERROR exception and replaced it
with more descriptive codes.

Fixed a problem where an exception would occur if an ASL Field was
defined with no named Field Units underneath it (used by some
index fields).

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  68.8K Code,   7.1K Data,   75.9K Total
    Debug Version:     142.9K Code,  58.4K Data,  201.3K Total
  Current Release:
    Non-Debug Version:  69.3K Code,   7.4K Data,   76.7K Total
    Debug Version:     143.8K Code,  58.8K Data,  202.6K Total
```

2) Linux

Much work done on ACPI init (MADT and PCI IRQ routing support).
(Paul D. and Dominik Brodowski)

Fix PCI IRQ-related panic on boot (Sam Revitch)

Set BM_ARB_DIS when entering a sleep state (Ducrot Bruno)

Fix "MHz" typo (Dominik Brodowski)

Fix RTC year 2000 issue (Dominik Brodowski)

Preclude multiple button proc entries (Eric Brunet)

Moved arch-specific code out of include/platform/aclinux.h

3) iASL Compiler Version X2044:

Implemented error checking for the string used in the EISAID macro
(Usually used in the definition of the _HID object.) The code now
strictly enforces the PnP format - exactly 7 characters, 3
uppercase letters and 4 hex digits.

If a raw string is used in the definition of the _HID object
(instead of the EISAID macro), the string must contain all
alphanumeric characters (e.g., "*PNP0011" is not allowed because
of the asterisk.)

Implemented checking for invalid use of ACPI reserved names for
most of the name creation operators (Name, Device, Event, Mutex,
OperationRegion, PowerResource, Processor, and ThermalZone.)
Previously, this check was only performed for control methods.

Implemented an additional check on the Name operator to emit an
error if a reserved name that must be implemented in ASL as a
control method is used. We know that a reserved name must be a
method if it is defined with input arguments.

The warning emitted when a namespace object reference is not found
during the cross reference phase has been changed into an error.
The "External" directive should be used for names defined in other
modules.

4) Tools and Utilities

The 16-bit tools (adump16 and aexec16) have been regenerated and
tested.

Fixed a problem with the output of both acpidump and adump16 where
the indentation of closing parentheses and brackets was not

aligned properly with the parent block.

---
03 May 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020503:

Added support a new OSL interface that allows the host operating

system software to override the DSDT found in the firmware -
AcpiOsTableOverride. With this interface, the OSL can examine the
version of the firmware DSDT and replace it with a different one
if desired.

Added new external interfaces for accessing ACPI registers from
device drivers and other system software - AcpiGetRegister and
AcpiSetRegister. This was simply an externalization of the
existing AcpiHwBitRegister interfaces.

Fixed a regression introduced in the previous build where the
ASL/AML CreateField operator always returned an error,
"destination must be a NS Node".

Extended the maximum time (before failure) to successfully enable
ACPI mode to 3 seconds.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release
    Non-Debug Version:  68.5K Code,   7.0K Data,   75.5K Total
    Debug Version:     142.4K Code,  58.3K Data,  200.7K Total
  Current Release:
    Non-Debug Version:  68.8K Code,   7.1K Data,   75.9K Total
    Debug Version:     142.9K Code,  58.4K Data,  201.3K Total
```

2) Linux

Enhanced ACPI init code for SMP. We are now fully MPS and $PIR-
free. While 3 out of 4 of our in-house systems work fine, the last
one still hangs when testing the LAPIC timer.

Renamed many files in 2.5 kernel release to omit "acpi_" from the name.

Added warning on boot for Presario 711FR.

Sleep improvements (Pavel Machek)

ACPI can now be built without CONFIG_PCI enabled.

IA64: Fixed memory map functions (JI Lee)

3) iASL Compiler Version X2043:

Added support to allow the compiler to be integrated into the MS VC++ development environment for one-button compilation of single files or entire projects -- with error-to-source-line mapping.

Implemented support for compile-time constant folding for the Type3, Type4, and Type5 opcodes first defined in the ACPI 2.0 specification. This allows the ASL writer to use expressions instead of Integer/Buffer/String constants in terms that must evaluate to constants at compile time and will also simplify the emitted AML in any such sub-expressions that can be folded (evaluated at compile-time.) This increases the size of the compiler significantly because a portion of the ACPI CA AML interpreter is included within the compiler in order to pre-evaluate constant expressions.

Fixed a problem with the "Unicode" ASL macro that caused the compiler to fault. (This macro is used in conjunction with the _STR reserved name.)

Implemented an AML opcode optimization to use the Zero, One, and Ones opcodes where possible to further reduce the size of integer constants and thus reduce the overall size of the generated AML code.

Implemented error checking for new reserved terms for ACPI version 2.0A.

Implemented the -qr option to display the current list of ACPI reserved names known to the compiler.

Implemented the -qc option to display the current list of ASL operators that are allowed within constant expressions and can therefore be folded at compile time if the operands are constants.

4) Documentation

Updated the Programmer's Reference for new interfaces, data types, and memory allocation model options.

Updated the iASL Compiler User Reference to apply new format and add information about new features and options.

---

19 April 2002. Summary of changes for this release.

1) ACPI CA Core Subsystem Version 20020419:

The source code base for the Core Subsystem has been completely cleaned with PC-lint (FlexLint) for both 32-bit and 64-bit versions. The Lint option files used are included in the

/acpi/generate/lint directory.

Implemented enhanced status/error checking across the entire Hardware manager subsystem. Any hardware errors (reported from the OSL) are now bubbled up and will abort a running control method.

Fixed a problem where the per-ACPI-table integer width (32 or 64) was stored only with control method nodes, causing a fault when non-control method code was executed during table loading. The solution implemented uses a global variable to indicate table width across the entire ACPI subsystem. Therefore, ACPI CA does not support mixed integer widths across different ACPI tables (DSDT, SSDT).

Fixed a problem where NULL extended fields (X fields) in an ACPI 2.0 ACPI FADT caused the table load to fail. Although the existing ACPI specification is a bit fuzzy on this topic, the new behavior is to fall back on a ACPI 1.0 field if the corresponding ACPI 2.0 X field is zero (even though the table revision indicates a full ACPI 2.0 table.) The ACPI specification will be updated to clarify this issue.

Fixed a problem with the SystemMemory operation region handler where memory was always accessed byte-wise even if the AML-specified access width was larger than a byte. This caused problems on systems with memory-mapped I/O. Memory is now accessed with the width specified. On systems that do not support non-aligned transfers, a check is made to guarantee proper address alignment before proceeding in order to avoid an AML-caused alignment fault within the kernel.

Fixed a problem with the ExtendedIrq resource where only one byte of the 4-byte Irq field was extracted.

Fixed the AcpiExDigitsNeeded() procedure to support _UID. This function was out of date and required a rewrite.

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release
    Non-Debug Version:  66.6K Code,   6.5K Data,   73.1K Total
    Debug Version:     139.8K Code,  57.4K Data,  197.2K Total
  Current Release:
    Non-Debug Version:  68.5K Code,   7.0K Data,   75.5K Total
    Debug Version:     142.4K Code,  58.3K Data,  200.7K Total
```

2) Linux

PCI IRQ routing fixes (Dominik Brodowski)

3) iASL Compiler Version X2042:

Implemented an additional compile-time error check for a field unit whose size + minimum access width would cause a run-time

access beyond the end-of-region. Previously, only the field size
itself was checked.

The Core subsystem and iASL compiler now share a common parse
object in preparation for compile-time evaluation of the type
3/4/5 ASL operators.

Summary of changes for this release: 03_29_02

1) ACPI CA Core Subsystem Version 20020329:

Implemented support for late evaluation of TermArg operands to
Buffer and Package objects. This allows complex expressions to be
used in the declarations of these object types.

Fixed an ACPI 1.0 compatibility issue when reading Fields. In ACPI
1.0, if the field was larger than 32 bits, it was returned as a
buffer - otherwise it was returned as an integer. In ACPI 2.0,
the field is returned as a buffer only if the field is larger than
64 bits. The TableRevision is now considered when making this
conversion to avoid incompatibility with existing ASL code.

Implemented logical addressing for AcpiOsGetRootPointer. This
allows an RSDP with either a logical or physical address. With
this support, the host OS can now override all ACPI tables with
one logical RSDP. Includes implementation of "typed" pointer
support to allow a common data type for both physical and logical
pointers internally. This required a change to the
AcpiOsGetRootPointer interface.

Implemented the use of ACPI 2.0 Generic Address Structures for all
GPE, Fixed Event, and PM Timer I/O. This allows the use of memory
mapped I/O for these ACPI features.

Initialization now ignores not only non-required tables (All
tables other than the FADT, FACS, DSDT, and SSDTs), but also does
not validate the table headers of unrecognized tables.

Fixed a problem where a notify handler could only be
installed/removed on an object of type Device. All "notify"

objects are now supported -- Devices, Processor, Power, and
Thermal.

Removed most verbosity from the ACPI_DB_INFO debug level. Only
critical information is returned when this debug level is enabled.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

  Previous Release
    Non-Debug Version: 65.4K Code,   6.2K Data,   71.6K Total
    Debug Version: 138.0K Code, 56.6K Data, 194.6K Total
  Current Release:
    Non-Debug Version: 66.6K Code,   6.5K Data,   73.1K Total
    Debug Version: 139.8K Code, 57.4K Data, 197.2K Total

2) Linux:

The processor driver (acpi_processor.c) now fully supports ACPI
2.0-based processor performance control (e.g. Intel(R)
SpeedStep(TM) technology) Note that older laptops that only have
the Intel "applet" interface are not supported through this. The
'limit' and 'performance' interface (/proc) are fully functional.
[Note that basic policy for controlling performance state
transitions will be included in the next version of ospmd.] The
idle handler was modified to more aggressively use C2, and PIIX4
errata handling underwent a complete overhaul (big thanks to
Dominik Brodowski).

Added support for ACPI-PCI device binding (acpi_pci_root.c). _ADR-
based devices in the ACPI namespace are now dynamically bound
(associated) with their PCI counterparts (e.g. PCI1->01:00.0).
This allows, among other things, ACPI to resolve bus numbers for
subordinate PCI bridges.

Enhanced PCI IRQ routing to get the proper bus number for _PRT
entries defined underneath PCI bridges.

Added IBM 600E to bad bios list due to invalid _ADR value for
PIIX4 PCI-ISA bridge, resulting in improper PCI IRQ routing.

In the process of adding full MADT support (e.g. IOAPIC) for IA32
(acpi.c, mpparse.c) -- stay tuned.

Added back visual differentiation between fixed-feature and
control-method buttons in dmesg. Buttons are also subtyped (e.g.
button/power/PWRF) to simplify button identification.

We no longer use -Wno-unused when compiling debug. Please ignore
any "_THIS_MODULE defined but not used" messages.

Can now shut down the system using "magic sysrq" key.

3) iASL Compiler version 2041:

Fixed a problem where conversion errors for hex/octal/decimal
constants were not reported.

Implemented a fix for the General Register template Address field.
This field was 8 bits when it should be 64.

Fixed a problem where errors/warnings were no longer being emitted
within the listing output file.

Implemented the ACPI 2.0A restriction on ACPI Table Signatures to
exactly 4 characters, alphanumeric only.

Summary of changes for this release: 03_08_02

1) ACPI CA Core Subsystem Version 20020308:

Fixed a problem with AML Fields where the use of the "AccessAny"
keyword could cause an interpreter error due to attempting to read
or write beyond the end of the parent Operation Region.

Fixed a problem in the SystemMemory Operation Region handler where

an attempt was made to map memory beyond the end of the region.
This was the root cause of the "AE_ERROR" and "AE_NO_MEMORY"
errors on some Linux systems.

Fixed a problem where the interpreter/namespace "search to root"
algorithm was not functioning for some object types. Relaxed the
internal restriction on the search to allow upsearches for all
external object types as well as most internal types.

2) Linux:

We now use safe_halt() macro versus individual calls to sti | hlt.

Writing to the processor limit interface should now work. "echo 1"
will increase the limit, 2 will decrease, and 0 will reset to the

default.

3) ASL compiler:

Fixed segfault on Linux version.

Summary of changes for this release: 02_25_02

1) ACPI CA Core Subsystem:

Fixed a problem where the GPE bit masks were not initialized
properly, causing erratic GPE behavior.

Implemented limited support for multiple calling conventions. The
code can be generated with either the VPL (variable parameter
list, or "C") convention, or the FPL (fixed parameter list, or
"Pascal") convention. The core subsystem is about 3.4% smaller
when generated with FPL.

2) Linux

Re-add some /proc/acpi/event functionality that was lost during
the rewrite

Resolved issue with /proc events for fixed-feature buttons showing
up as the system device.

Fixed checks on C2/C3 latencies to be inclusive of maximum values.

Replaced AE_ERRORs in acpi_osl.c with more specific error codes.

Changed ACPI PRT option from "pci=noacpi-routing" to "pci=noacpi"

Fixed limit interface & usage to fix bugs with passive cooling
hysterisis.

Restructured PRT support.

Summary of changes for this label: 02_14_02

1) ACPI CA Core Subsystem:

Implemented support in AcpiLoadTable to allow loading of FACS and
FADT tables.

Suport for the now-obsolete interim 0.71 64-bit ACPI tables has
been removed. All 64-bit platforms should be migrated to the ACPI
2.0 tables. The actbl71.h header has been removed from the source
tree.

All C macros defined within the subsystem have been prefixed with
"ACPI_" to avoid collision with other system include files.

Removed the return value for the two AcpiOsPrint interfaces, since
it is never used and causes lint warnings for ignoring the return
value.

Added error checking to all internal mutex acquire and release
calls. Although a failure from one of these interfaces is
probably a fatal system error, these checks will cause the
immediate abort of the currently executing method or interface.

Fixed a problem where the AcpiSetCurrentResources interface could
fault. This was a side effect of the deployment of the new memory
allocation model.

Fixed a couple of problems with the Global Lock support introduced
in the last major build. The "common" (1.0/2.0) internal FACS was
being overwritten with the FACS signature and clobbering the
Global Lock pointer. Also, the actual firmware FACS was being
unmapped after construction of the "common" FACS, preventing
access to the actual Global Lock field within it. The "common"
internal FACS is no longer installed as an actual ACPI table; it
is used simply as a global.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release (02_07_01)
    Non-Debug Version:  65.2K Code,   6.2K Data,   71.4K Total
    Debug Version:     136.9K Code,  56.4K Data,  193.3K Total
  Current Release:
    Non-Debug Version:  65.4K Code,   6.2K Data,   71.6K Total
    Debug Version:     138.0K Code,  56.6K Data,  194.6K Total
```

2) Linux

Updated Linux-specific code for core macro and OSL interface
changes described above.

Improved /proc/acpi/event. It now can be opened only once and has
proper poll functionality.

Fixed and restructured power management (acpi_bus).

Only create /proc "view by type" when devices of that class exist.

Fixed "charging/discharging" bug (and others) in acpi_battery.

Improved thermal zone code.

12895 3) ASL Compiler, version X2039:

12898 Implemented the new compiler restriction on ASL String hex/octal
12899 escapes to non-null, ASCII values. An error results if an invalid
12900 value is used. (This will require an ACPI 2.0 specification
12901 change.)

12903 AML object labels that are output to the optional C and ASM source
12904 are now prefixed with both the ACPI table signature and table ID
12905 to help guarantee uniqueness within a large BIOS project.

12908 --
12909 Summary of changes for this label: 02_01_02

12911 1) ACPI CA Core Subsystem:

12913 ACPI 2.0 support is complete in the entire Core Subsystem and the
12914 ASL compiler. All new ACPI 2.0 operators are implemented and all
12915 other changes for ACPI 2.0 support are complete. With
12916 simultaneous code and data optimizations throughout the subsystem,
12917 ACPI 2.0 support has been implemented with almost no additional
12918 cost in terms of code and data size.

12920 Implemented a new mechanism for allocation of return buffers. If
12921 the buffer length is set to ACPI_ALLOCATE_BUFFER, the buffer will
12922 be allocated on behalf of the caller. Consolidated all return
12923 buffer validation and allocation to a common procedure. Return
12924 buffers will be allocated via the primary OSL allocation interface
12925 since it appears that a separate pool is not needed by most users.
12926 If a separate pool is required for these buffers, the caller can
12927 still use the original mechanism and pre-allocate the buffer(s).

12929 Implemented support for string operands within the DerefOf
12930 operator.

12932 Restructured the Hardware and Event managers to be table driven,
12933 simplifying the source code and reducing the amount of generated
12934 code.

12936 Split the common read/write low-level ACPI register bitfield
12937 procedure into a separate read and write, simplifying the code
12938 considerably.

12940 Obsoleted the AcpiOsCallocate OSL interface. This interface was
12941 used only a handful of times and didn't have enough critical mass
12942 for a separate interface. Replaced with a common calloc procedure
12943 in the core.

12945 Fixed a reported problem with the GPE number mapping mechanism
12946 that allows GPE1 numbers to be non-contiguous with GPE0.
12947 Reorganized the GPE information and shrunk a large array that was
12948 originally large enough to hold info for all possible GPEs (256)
12949 to simply large enough to hold all GPEs up to the largest GPE
12950 number on the machine.

12952 Fixed a reported problem with resource structure alignment on 64-
12953 bit platforms.

12955 Changed the AcpiEnableEvent and AcpiDisableEvent external
12956 interfaces to not require any flags for the common case of
12957 enabling/disabling a GPE.

12959 Implemented support to allow a "Notify" on a Processor object.

12961 Most TBDs in comments within the source code have been resolved
12962 and eliminated.

12965 Fixed a problem in the interpreter where a standalone parent
12966 prefix (^) was not handled correctly in the interpreter and
12967 debugger.

12969 Removed obsolete and unnecessary GPE save/restore code.

12971 Implemented Field support in the ASL Load operator. This allows a
12972 table to be loaded from a named field, in addition to loading a
12973 table directly from an Operation Region.

12975 Implemented timeout and handle support in the external Global Lock
12976 interfaces.

12978 Fixed a problem in the AcpiDump utility where pathnames were no
12979 longer being generated correctly during the dump of named objects.

12981 Modified the AML debugger to give a full display of if/while
12982 predicates instead of just one AML opcode at a time. (The
12983 predicate can have several nested ASL statements.) The old method
12984 was confusing during single stepping.

12986 Code and Data Size: Current core subsystem library sizes are shown
12987 below. These are the code and data sizes for the acpica.lib
12988 produced by the Microsoft Visual C++ 6.0 compiler, and these
12989 values do not include any ACPI driver or OSPM code. The debug
12990 version of the code includes the debug output trace mechanism and
12991 has a larger code and data size. Note that these values will vary
12992 depending on the efficiency of the compiler and the compiler
12993 options used during generation.

12995 Previous Release (12_18_01)
12996 Non-Debug Version: 66.1K Code, 5.5K Data, 71.6K Total
12997 Debug Version: 138.3K Code, 55.9K Data, 194.2K Total
12998 Current Release:
12999 Non-Debug Version: 65.2K Code, 6.2K Data, 71.4K Total
13000 Debug Version: 136.9K Code, 56.4K Data, 193.3K Total

13002 2) Linux

13004 Implemented fix for PIIX reverse throttling errata (Processor
13005 driver)

13007 Added new Limit interface (Processor and Thermal drivers)

13009 New thermal policy (Thermal driver)

13011 Many updates to /proc

13013 Battery "low" event support (Battery driver)

13015 Supports ACPI PCI IRQ routing (PCI Link and PCI root drivers)

13017 IA32 - IA64 initialization unification, no longer experimental

13019 Menuconfig options redesigned

13021 3) ASL Compiler, version X2037:

13023 Implemented several new output features to simplify integration of
13024 AML code into firmware: 1) Output the AML in C source code with

labels for each named ASL object. The original ASL source code is interleaved as C comments. 2) Output the AML in ASM source code with labels and interleaved ASL source. 3) Output the AML in raw hex table form, in either C or ASM.

Implemented support for optional string parameters to the LoadTable operator.

Completed support for embedded escape sequences within string literals. The compiler now supports all single character escapes as well as the Octal and Hex escapes. Note: the insertion of a null byte into a string literal (via the hex/octal escape) causes the string to be immediately terminated. A warning is issued.

Fixed a problem where incorrect AML was generated for the case where an ASL namepath consists of a single parent prefix (

) with no trailing name segments.

The compiler has been successfully generated with a 64-bit C compiler.

Summary of changes for this label: 12_18_01

1) Linux

Enhanced blacklist with reason and severity fields. Any table's signature may now be used to identify a blacklisted system.

Call _PIC control method to inform the firmware which interrupt model the OS is using. Turn on any disabled link devices.

Cleaned up busmgr /proc error handling (Andreas Dilger)

2) ACPI CA Core Subsystem:

Implemented ACPI 2.0 semantics for the "Break" operator (Exit from while loop)

Completed implementation of the ACPI 2.0 "Continue", "ConcatenateResTemplate", "DataTableRegion", and "LoadTable" operators. All new ACPI 2.0 operators are now implemented in both the ASL compiler and the AML interpreter. The only remaining ACPI 2.0 task is support for the String data type in the DerefOf operator. Fixed a problem with AcquireMutex where the status code was lost if the caller had to actually wait for the mutex.

Increased the maximum ASL Field size from 64K bits to 4G bits.

Completed implementation of the external Global Lock interfaces -- AcpiAcquireGlobalLock and AcpiReleaseGlobalLock. The Timeout and Handler parameters were added.

Completed another pass at removing warnings and issues when compiling with 64-bit compilers. The code now compiles cleanly with the Intel 64-bit C/C++ compiler. Most notably, the pointer add and subtract (diff) macros have changed considerably.

Created and deployed a new ACPI_SIZE type that is 64-bits wide on 64-bit platforms, 32-bits on all others. This type is used wherever memory allocation and/or the C sizeof() operator is used,

and affects the OSL memory allocation interfaces AcpiOsAllocate and AcpiOsCallocate.

Implemented sticky user breakpoints in the AML debugger.

Code and Data Size: Current core subsystem library sizes are shown below. These are the code and data sizes for the acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and these values do not include any ACPI driver or OSPM code. The debug version of the code includes the debug output trace mechanism and has a larger code and data size. Note that these values will vary depending on the efficiency of the compiler and the compiler options used during generation.

```
  Previous Release (12_05_01)
     Non-Debug Version:  64.7K Code,   5.3K Data,   70.0K Total
     Debug Version:     136.2K Code,  55.6K Data,  191.8K Total
   Current Release:
     Non-Debug Version:  66.1K Code,   5.5K Data,   71.6K Total
     Debug Version:     138.3K Code,  55.9K Data,  194.2K Total
```

3) ASL Compiler, version X2034:

Now checks for (and generates an error if detected) the use of a Break or Continue statement without an enclosing While statement.

Successfully generated the compiler with the Intel 64-bit C compiler.

Summary of changes for this label: 12_05_01

1) ACPI CA Core Subsystem:

The ACPI 2.0 CopyObject operator is fully implemented. This operator creates a new copy of an object (and is also used to bypass the "implicit conversion" mechanism of the Store operator.)

The ACPI 2.0 semantics for the SizeOf operator are fully implemented. The change is that performing a SizeOf on a reference object causes an automatic dereference of the object to tha actual value before the size is evaluated. This behavior was undefined in ACPI 1.0.

The ACPI 2.0 semantics for the Extended IRQ resource descriptor have been implemented. The interrupt polarity and mode are now independently set.

Fixed a problem where ASL Constants (Zero, One, Ones, Revision) appearing in Package objects were not properly converted to integers when the internal Package was converted to an external object (via the AcpiEvaluateObject interface.)

Fixed a problem with the namespace object deletion mechanism for objects created by control methods. There were two parts to this problem: 1) Objects created during the initialization phase method parse were not being deleted, and 2) The object owner ID mechanism to track objects was broken.

Fixed a problem where the use of the ASL Scope operator within a control method would result in an invalid opcode exception.

Fixed a problem introduced in the previous label where the buffer length required for the _PRT structure was not being returned correctly.

Code and Data Size: Current core subsystem library sizes are shown
below. These are the code and data sizes for the acpica.lib
produced by the Microsoft Visual C++ 6.0 compiler, and these
values do not include any ACPI driver or OSPM code. The debug
version of the code includes the debug output trace mechanism and
has a larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
  Previous Release (11_20_01)
     Non-Debug Version:  64.1K Code,   5.3K Data,   69.4K Total
     Debug Version:     135.1K Code,  55.4K Data,  190.5K Total

  Current Release:
     Non-Debug Version:  64.7K Code,   5.3K Data,   70.0K Total
     Debug Version:     136.2K Code,  55.6K Data,  191.8K Total
```

2) Linux:

Updated all files to apply cleanly against 2.4.16.

Added basic PCI Interrupt Routing Table (PRT) support for IA32
(acpi_pci.c), and unified the PRT code for IA32 and IA64. This
version supports both static and dyanmic PRT entries, but dynamic
entries are treated as if they were static (not yet
reconfigurable). Architecture- specific code to use this data is
absent on IA32 but should be available shortly.

Changed the initialization sequence to start the ACPI interpreter
(acpi_init) prior to initialization of the PCI driver (pci_init)
in init/main.c. This ordering is required to support PRT and
facilitate other (future) enhancement. A side effect is that the
ACPI bus driver and certain device drivers can no longer be loaded
as modules.

Modified the 'make menuconfig' options to allow PCI Interrupt
Routing support to be included without the ACPI Bus and other
device drivers.

3) ASL Compiler, version X2033:

Fixed some issues with the use of the new CopyObject and
DataTableRegion operators. Both are fully functional.

```
 ----------------------------------------
```

Summary of changes for this label: 11_20_01

20 November 2001. Summary of changes for this release.

1) ACPI CA Core Subsystem:

Updated Index support to match ACPI 2.0 semantics. Storing a
Integer, String, or Buffer to an Index of a Buffer will store only
the least-significant byte of the source to the Indexed buffer
byte. Multiple writes are not performed.

Fixed a problem where the access type used in an AccessAs ASL
operator was not recorded correctly into the field object.

Fixed a problem where ASL Event objects were created in a
signalled state. Events are now created in an unsignalled state.

The internal object cache is now purged after table loading and
initialization to reduce the use of dynamic kernel memory -- on
the assumption that object use is greatest during the parse phase

of the entire table (versus the run-time use of individual control
methods.)

ACPI 2.0 variable-length packages are now fully operational.

Code and Data Size: Code and Data optimizations have permitted new
feature development with an actual reduction in the library size.
Current core subsystem library sizes are shown below. These are
the code and data sizes for the acpica.lib produced by the
Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code
includes the debug output trace mechanism and has a larger code
and data size. Note that these values will vary depending on the
efficiency of the compiler and the compiler options used during
generation.

```
  Previous Release (11_09_01):
     Non-Debug Version:  63.7K Code,   5.2K Data,   68.9K Total
     Debug Version:     134.5K Code,  55.4K Data,  189.9K Total

  Current Release:
     Non-Debug Version:  64.1K Code,   5.3K Data,   69.4K Total
     Debug Version:     135.1K Code,  55.4K Data,  190.5K Total
```

2) Linux:

Enhanced the ACPI boot-time initialization code to allow the use
of Local APIC tables for processor enumeration on IA-32, and to
pave the way for a fully MPS-free boot (on SMP systems) in the
near future. This functionality replaces
arch/i386/kernel/acpitables.c, which was introduced in an earlier
2.4.15-preX release. To enable this feature you must add
"acpi_boot=on" to the kernel command line -- see the help entry
for CONFIG_ACPI_BOOT for more information. An IA-64 release is in
the works...

Restructured the configuration options to allow boot-time table
parsing support without inclusion of the ACPI Interpreter (and
other) code.

NOTE: This release does not include fixes for the reported events,
power-down, and thermal passive cooling issues (coming soon).

3) ASL Compiler:

Added additional typechecking for Fields within restricted access
Operation Regions. All fields within EC and CMOS regions must be
declared with ByteAcc. All fields withing SMBus regions must be
declared with the BufferAcc access type.

Fixed a problem where the listing file output of control methods
no longer interleaved the actual AML code with the ASL source
code.

```
----------------------------------------
```

Summary of changes for this label: 11_09_01

1) ACPI CA Core Subsystem:

Implemented ACPI 2.0-defined support for writes to fields with a
Buffer, String, or Integer source operand that is smaller than the
target field. In these cases, the source operand is zero-extended
to fill the target field.

Fixed a problem where a Field starting bit offset (within the
parent operation region) was calculated incorrectly if the

alignment of the field differed from the access width. This
affected CreateWordField, CreateDwordField, CreateQwordField, and
possibly other fields that use the "AccessAny" keyword.

Fixed a problem introduced in the 11_02_01 release where indirect
stores through method arguments did not operate correctly.

2) Linux:

Implemented boot-time ACPI table parsing support
(CONFIG_ACPI_BOOT) for IA32 and IA64 UP/SMP systems. This code
facilitates the use of ACPI tables (e.g. MADT, SRAT) rather than
legacy BIOS interfaces (e.g. MPS) for the configuration of system
processors, memory, and interrupts during setup_arch(). Note that
this patch does not include the required architecture-specific
changes required to apply this information -- subsequent patches
will be posted for both IA32 and IA64 to achieve this.

Added low-level sleep support for IA32 platforms, courtesy of Pat
Mochel. This allows IA32 systems to transition to/from various
sleeping states (e.g. S1, S3), although the lack of a centralized
driver model and power-manageable drivers will prevent its
(successful) use on most systems.

Revamped the ACPI 'menuconfig' layout: created new "ACPI Support"
submenu, unified IA32 and IA64 options, added new "Boot using ACPI
tables" option, etc.

Increased the default timeout for the EC driver from 1ms to 10ms
(1000 cycles of 10us) to try to address AE_TIME errors during EC
transactions.

Summary of changes for this label: 11_02_01

1) ACPI CA Core Subsystem:

ACPI 2.0 Support: Implemented ACPI 2.0 64-bit Field access
(QWordAcc keyword). All ACPI 2.0 64-bit support is now
implemented.

OSL Interfaces: Several of the OSL (AcpiOs*) interfaces required
changes to support ACPI 2.0 Qword field access. Read/Write
PciConfiguration(), Read/Write Memory(), and Read/Write Port() now
accept an ACPI_INTEGER (64 bits) as the value parameter. Also,
the value parameter for the address space handler interface is now
an ACPI_INTEGER. OSL implementations of these interfaces must now
handle the case where the Width parameter is 64.

Index Fields: Fixed a problem where unaligned bit assembly and
disassembly for IndexFields was not supported correctly.

Index and Bank Fields: Nested Index and Bank Fields are now
supported. During field access, a check is performed to ensure
that the value written to an Index or Bank register is not out of
the range of the register. The Index (or Bank) register is
written before each access to the field data. Future support will
include allowing individual IndexFields to be wider than the
DataRegister width.

Fields: Fixed a problem where the AML interpreter was incorrectly
attempting to write beyond the end of a Field/OpRegion. This was

a boundary case that occurred when a DWORD field was written to a
BYTE access OpRegion, forcing multiple writes and causing the
interpreter to write one datum too many.

Fields: Fixed a problem with Field/OpRegion access where the
starting bit address of a field was incorrectly calculated if the
current access type was wider than a byte (WordAcc, DwordAcc, or
QwordAcc).

Fields: Fixed a problem where forward references to individual
FieldUnits (individual Field names within a Field definition) were
not resolved during the AML table load.

Fields: Fixed a problem where forward references from a Field
definition to the parent Operation Region definition were not
resolved during the AML table load.

Fields: Duplicate FieldUnit names within a scope are now detected
during AML table load.

Acpi Interfaces: Fixed a problem where the AcpiGetName() interface
returned an incorrect name for the root node.

Code and Data Size: Code and Data optimizations have permitted new
feature development with an actual reduction in the library size.
Current core subsystem library sizes are shown below. These are
the code and data sizes for the acpica.lib produced by the
Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code
includes the debug output trace mechanism and has a larger code
and data size. Note that these values will vary depending on the
efficiency of the compiler and the compiler options used during
generation.

  Previous Release (10_18_01):
     Non-Debug Version: 63.9K Code,   5.1K Data,   69.0K Total
     Debug Version: 136.7K Code, 57.4K Data, 194.2K Total

  Current Release:
     Non-Debug Version: 63.7K Code,   5.2K Data,   68.9K Total
     Debug Version: 134.5K Code, 55.4K Data, 189.9K Total

 2) Linux:

Improved /proc processor output (Pavel Machek) Re-added
MODULE_LICENSE("GPL") to all modules.

 3) ASL Compiler version X2030:

Duplicate FieldUnit names within a scope are now detected and
flagged as errors.

 4) Documentation:

Programmer Reference updated to reflect OSL and address space
handler interface changes described above.

Summary of changes for this label: 10_18_01

ACPI CA Core Subsystem:

Fixed a problem with the internal object reference count mechanism
that occasionally caused premature object deletion. This resolves
all of the outstanding problem reports where an object is deleted
in the middle of an interpreter evaluation. Although this problem

only showed up in rather obscure cases, the solution to the
problem involved an adjustment of all reference counts involving
objects attached to namespace nodes.

Fixed a problem with Field support in the interpreter where
writing to an aligned field whose length is an exact multiple (2
or greater) of the field access granularity would cause an attempt
to write beyond the end of the field.

The top level AML opcode execution functions within the
interpreter have been renamed with a more meaningful and
consistent naming convention. The modules exmonad.c and
exdyadic.c were eliminated. New modules are exoparg1.c,
exoparg2.c, exoparg3.c, and exoparg6.c.

Support for the ACPI 2.0 "Mid" ASL operator has been implemented.

Fixed a problem where the AML debugger was causing some internal
objects to not be deleted during subsystem termination.

Fixed a problem with the external AcpiEvaluateObject interface
where the subsystem would fault if the named object to be
evaluated refered to a constant such as Zero, Ones, etc.

Fixed a problem with IndexFields and BankFields where the
subsystem would fault if the index, data, or bank registers were
not defined in the same scope as the field itself.

Added printf format string checking for compilers that support
this feature. Corrected more than 50 instances of issues with
format specifiers within invocations of ACPI_DEBUG_PRINT
throughout the core subsystem code.

The ASL "Revision" operator now returns the ACPI support level
implemented in the core - the value "2" since the ACPI 2.0 support
is more than 50% implemented.

Enhanced the output of the AML debugger "dump namespace" command
to output in a more human-readable form.

Current core subsystem library code sizes are shown below. These

are the code and data sizes for the acpica.lib produced by the
Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code
includes the full debug trace mechanism -- leading to a much

larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

```
     Previous Label (09_20_01):
     Non-Debug Version:    65K Code,     5K Data,     70K Total
     Debug Version:       138K Code,    58K Data,    196K Total

     This Label:

     Non-Debug Version:  63.9K Code,   5.1K Data,   69.0K Total
     Debug Version:     136.7K Code,  57.4K Data,  194.2K Total
```

Linux:

Implemented a "Bad BIOS Blacklist" to track machines that have
known ASL/AML problems.

Enhanced the /proc interface for the thermal zone driver and added

support for _HOT (the critical suspend trip point). The 'info'
file now includes threshold/policy information, and allows setting
of _SCP (cooling preference) and _TZP (polling frequency) values
to the 'info' file. Examples: "echo tzp=5 > info" sets the polling
frequency to 5 seconds, and "echo scp=1 > info" sets the cooling
preference to the passive/quiet mode (if supported by the ASL).

Implemented a workaround for a gcc bug that resuted in an OOPs
when loading the control method battery driver.

Summary of changes for this label: 09_20_01

 ACPI CA Core Subsystem:

The AcpiEnableEvent and AcpiDisableEvent interfaces have been
modified to allow individual GPE levels to be flagged as wake-
enabled (i.e., these GPEs are to remain enabled when the platform
sleeps.)

The AcpiEnterSleepState and AcpiLeaveSleepState interfaces now
support wake-enabled GPEs. This means that upon entering the
sleep state, all GPEs that are not wake-enabled are disabled.
When leaving the sleep state, these GPEs are reenabled.

A local double-precision divide/modulo module has been added to
enhance portability to OS kernels where a 64-bit math library is
not available. The new module is "utmath.c".

Several optimizations have been made to reduce the use of CPU
stack. Originally over 2K, the maximum stack usage is now below
2K at 1860 bytes (1.82k)

Fixed a problem with the AcpiGetFirmwareTable interface where the
root table pointer was not mapped into a logical address properly.

Fixed a problem where a NULL pointer was being dereferenced in the
interpreter code for the ASL Notify operator.

Fixed a problem where the use of the ASL Revision operator
returned an error. This operator now returns the current version
of the ACPI CA core subsystem.

Fixed a problem where objects passed as control method parameters
to AcpiEvaluateObject were always deleted at method termination.
However, these objects may end up being stored into the namespace
by the called method. The object reference count mechanism was
applied to these objects instead of a force delete.

Fixed a problem where static strings or buffers (contained in the
AML code) that are declared as package elements within the ASL
code could cause a fault because the interpreter would attempt to
delete them. These objects are now marked with the "static
object" flag to prevent any attempt to delete them.

Implemented an interpreter optimization to use operands directly
from the state object instead of extracting the operands to local
variables. This reduces stack use and code size, and improves
performance.

The module exxface.c was eliminated as it was an unnecessary extra
layer of code.

Current core subsystem library code sizes are shown below. These
are the code and data sizes for the acpica.lib produced by the
Microsoft Visual C++ 6.0 compiler, and these values do not include

any ACPI driver or OSPM code. The debug version of the code
includes the full debug trace mechanism -- leading to a much
larger code and data size. Note that these values will vary
depending on the efficiency of the compiler and the compiler
options used during generation.

  Non-Debug Version: 65K Code,   5K Data,   70K Total
(Previously 69K)   Debug Version: 138K Code, 58K Data, 196K
Total (Previously 195K)

Linux:

Support for ACPI 2.0 64-bit integers has been added.   All ACPI
Integer objects are now 64 bits wide

All Acpi data types and structures are now in lower case. Only
Acpi macros are upper case for differentiation.

 Documentation:

Changes to the external interfaces as described above.

Summary of changes for this label: 08_31_01

 ACPI CA Core Subsystem:

A bug with interpreter implementation of the ASL Divide operator
was found and fixed. The implicit function return value (not the
explicit store operands) was returning the remainder instead of
the quotient. This was a longstanding bug and it fixes several
known outstanding issues on various platforms.

The ACPI_DEBUG_PRINT and function trace entry/exit macros have
been further optimized for size. There are 700 invocations of the
DEBUG_PRINT macro alone, so each optimization reduces the size of
the debug version of the subsystem significantly.

A stack trace mechanism has been implemented. The maximum stack
usage is about 2K on 32-bit platforms. The debugger command "stat
stack" will display the current maximum stack usage.

All public symbols and global variables within the subsystem are
now prefixed with the string "Acpi". This keeps all of the
symbols grouped together in a kernel map, and avoids conflicts
with other kernel subsystems.

Most of the internal fixed lookup tables have been moved into the
code segment via the const operator.

Several enhancements have been made to the interpreter to both
reduce the code size and improve performance.

Current core subsystem library code sizes are shown below. These
are the code and data sizes for the acpica.lib produced by the
Microsoft Visual C++ 6.0 compiler, and these values do not include
any ACPI driver or OSPM code. The debug version of the code
includes the full debug trace mechanism which contains over 700
invocations of the DEBUG_PRINT macro, 500 function entry macro
invocations, and over 900 function exit macro invocations --
leading to a much larger code and data size. Note that these
values will vary depending on the efficiency of the compiler and
the compiler options used during generation.

        Non-Debug Version: 64K Code,   5K Data,   69K Total
Debug Version: 137K Code, 58K Data, 195K Total

 Linux:

Implemented wbinvd() macro, pending a kernel-wide definition.

Fixed /proc/acpi/event to handle poll() and short reads.

 ASL Compiler, version X2026:

Fixed a problem introduced in the previous label where the AML

code emitted for package objects produced packages with zero
length.

Summary of changes for this label: 08_16_01

ACPI CA Core Subsystem:

The following ACPI 2.0 ASL operators have been implemented in the
AML interpreter (These are already supported by the Intel ASL
compiler): ToDecimalString, ToHexString, ToString, ToInteger, and
ToBuffer. Support for 64-bit AML constants is implemented in the
AML parser, debugger, and disassembler.

The internal memory tracking mechanism (leak detection code) has
been upgraded to reduce the memory overhead (a separate tracking
block is no longer allocated for each memory allocation), and now
supports all of the internal object caches.

The data structures and code for the internal object caches have
been coelesced and optimized so that there is a single cache and
memory list data structure and a single group of functions that
implement generic cache management. This has reduced the code
size in both the debug and release versions of the subsystem.

The DEBUG_PRINT macro(s) have been optimized for size and replaced
by ACPI_DEBUG_PRINT. The syntax for this macro is slightly
different, because it generates a single call to an internal
function. This results in a savings of about 90 bytes per
invocation, resulting in an overall code and data savings of about
16% in the debug version of the subsystem.

 Linux:

Fixed C3 disk corruption problems and re-enabled C3 on supporting
machines.

Integrated low-level sleep code by Patrick Mochel.

Further tweaked source code Linuxization.

Other minor fixes.

 ASL Compiler:

Support for ACPI 2.0 variable length packages is fixed/completed.

Fixed a problem where the optional length parameter for the ACPI
2.0 ToString operator.

Fixed multiple extraneous error messages when a syntax error is
detected within the declaration line of a control method.

Summary of changes for this label: 07_17_01

ACPI CA Core Subsystem:

Added a new interface named AcpiGetFirmwareTable to obtain any ACPI table via the ACPI signature. The interface can be called at any time during kernel initialization, even before the kernel virtual memory manager is initialized and paging is enabled. This allows kernel subsystems to obtain ACPI tables very early, even before the ACPI CA subsystem is initialized.

Fixed a problem where Fields defined with the AnyAcc attribute could be resolved to the incorrect address under the following conditions: 1) the field width is larger than 8 bits and 2) the parent operation region is not defined on a DWORD boundary.

Fixed a problem where the interpreter is not being locked during namespace initialization (during execution of the _INI control methods), causing an error when an attempt is made to release it later.

ACPI 2.0 support in the AML Interpreter has begun and will be ongoing throughout the rest of this year. In this label, The Mod operator is implemented.

Added a new data type to contain full PCI addresses named ACPI_PCI_ID. This structure contains the PCI Segment, Bus, Device, and Function values.

Linux:

Enhanced the Linux version of the source code to change most capitalized ACPI type names to lowercase. For example, all instances of ACPI_STATUS are changed to acpi_status. This will result in a large diff, but the change is strictly cosmetic and aligns the CA code closer to the Linux coding standard.

OSL Interfaces:

The interfaces to the PCI configuration space have been changed to add the PCI Segment number and to split the single 32-bit combined DeviceFunction field into two 16-bit fields. This was accomplished by moving the four values that define an address in PCI configuration space (segment, bus, device, and function) to the new ACPI_PCI_ID structure.

The changes to the PCI configuration space interfaces led to a reexamination of the complete set of address space access interfaces for PCI, I/O, and Memory. The previously existing 18 interfaces have proven difficult to maintain (any small change must be propagated across at least 6 interfaces) and do not easily allow for future expansion to 64 bits if necessary. Also, on some systems, it would not be appropriate to demultiplex the access width (8, 16, 32,or 64) before calling the OSL if the corresponding native OS interfaces contain a similar access width parameter. For these reasons, the 18 address space interfaces have been replaced by these 6 new ones:

AcpiOsReadPciConfiguration
AcpiOsWritePciConfiguration
AcpiOsReadMemory
AcpiOsWriteMemory
AcpiOsReadPort
AcpiOsWritePort

Added a new interface named AcpiOsGetRootPointer to allow the OSL to perform the platform and/or OS-specific actions necessary to

obtain the ACPI RSDP table pointer. On IA-32 platforms, this interface will simply call down to the CA core to perform the low-memory search for the table. On IA-64, the RSDP is obtained from EFI. Migrating this interface to the OSL allows the CA core to

remain OS and platform independent.

Added a new interface named AcpiOsSignal to provide a generic "function code and pointer" interface for various miscellaneous signals and notifications that must be made to the host OS. The first such signals are intended to support the ASL Fatal and Breakpoint operators. In the latter case, the AcpiOsBreakpoint interface has been obsoleted.

The definition of the AcpiFormatException interface has been changed to simplify its use. The caller no longer must supply a buffer to the call; A pointer to a const string is now returned directly. This allows the call to be easily used in printf statements, etc. since the caller does not have to manage a local buffer.

ASL Compiler, Version X2025:

The ACPI 2.0 Switch/Case/Default operators have been implemented and are fully functional. They will work with all ACPI 1.0 interpreters, since the operators are simply translated to If/Else pairs.

The ACPI 2.0 ElseIf operator is implemented and will also work with 1.0 interpreters, for the same reason.

Implemented support for ACPI 2.0 variable-length packages. These packages have a separate opcode, and their size is determined by the interpreter at run-time.

Documentation The ACPI CA Programmer Reference has been updated to reflect the new interfaces and changes to existing interfaces.

Summary of changes for this label: 06_15_01

ACPI CA Core Subsystem:

Fixed a problem where a DWORD-accessed field within a Buffer object would get its byte address inadvertently rounded down to the nearest DWORD. Buffers are always Byte-accessible.

ASL Compiler, version X2024:

Fixed a problem where the Switch() operator would either fault or hang the compiler. Note however, that the AML code for this ACPI 2.0 operator is not yet implemented.

Compiler uses the new AcpiOsGetTimer interface to obtain compile timings.

Implementation of the CreateField operator automatically converts a reference to a named field within a resource descriptor from a byte offset to a bit offset if required.

Added some missing named fields from the resource descriptor support. These are the names that are automatically created by the compiler to reference fields within a descriptor. They are only valid at compile time and are not passed through to the AML interpreter.

Resource descriptor named fields are now typed as Integers and
subject to compile-time typechecking when used in expressions.

Summary of changes for this label: 05_18_01

 ACPI CA Core Subsystem:

Fixed a couple of problems in the Field support code where bits
from adjacent fields could be returned along with the proper field
bits. Restructured the field support code to improve performance,
readability and maintainability.

New DEBUG_PRINTP macro automatically inserts the procedure name
into the output, saving hundreds of copies of procedure name
strings within the source, shrinking the memory footprint of the
debug version of the core subsystem.

 Source Code Structure:

The source code directory tree was restructured to reflect the
current organization of the component architecture. Some files
and directories have been moved and/or renamed.

 Linux:

Fixed leaking kacpidpc processes.

Fixed queueing event data even when /proc/acpi/event is not
opened.

 ASL Compiler, version X2020:

Memory allocation performance enhancement - over 24X compile time
improvement on large ASL files. Parse nodes and namestring
buffers are now allocated from a large internal compiler buffer.

The temporary .SRC file is deleted unless the "-s" option is
specified

The "-d" debug output option now sends all output to the .DBG file
instead of the console.

"External" second parameter is now optional

"ElseIf" syntax now properly allows the predicate

Last operand to "Load" now recognized as a Target operand

Debug object can now be used anywhere as a normal object.

ResourceTemplate now returns an object of type BUFFER

EISAID now returns an object of type INTEGER

"Index" now works with a STRING operand

"LoadTable" now accepts optional parameters

"ToString" length parameter is now optional

"Interrupt (ResourceType," parse error fixed.

"Register" with a user-defined region space parse error fixed

Escaped backslash at the end of a string ("\\") scan/parse error
fixed

"Revision" is now an object of type INTEGER.

Summary of changes for this label: 05_02_01

Linux:

/proc/acpi/event now blocks properly.

Removed /proc/sys/acpi. You can still dump your DSDT from
/proc/acpi/dsdt.

 ACPI CA Core Subsystem:

Fixed a problem introduced in the previous label where some of the
"small" resource descriptor types were not recognized.

Improved error messages for the case where an ASL Field is outside
the range of the parent operation region.

 ASL Compiler, version X2018:

Added error detection for ASL Fields that extend beyond the length
of the parent operation region (only if the length of the region
is known at compile time.) This includes fields that have a
minimum access width that is smaller than the parent region, and
individual field units that are partially or entirely beyond the
extent of the parent.

Summary of changes for this label: 04_27_01

 ACPI CA Core Subsystem:

Fixed a problem where the namespace mutex could be released at the
wrong time during execution of AcpiRemoveAddressSpaceHandler.

Added optional thread ID output for debug traces, to simplify
debugging of multiple threads. Added context switch notification
when the debug code realizes that a different thread is now
executing ACPI code.

Some additional external data types have been prefixed with the
string "ACPI_" for consistency. This may effect existing code.
The data types affected are the external callback typedefs - e.g.,

WALK_CALLBACK becomes ACPI_WALK_CALLBACK.

 Linux:

Fixed an issue with the OSL semaphore implementation where a
thread was waking up with an error from receiving a SIGCHLD
signal.

Linux version of ACPI CA now uses the system C library for string
manipulation routines instead of a local implementation.

Cleaned up comments and removed TBDs.

```
13950  ASL Compiler, version X2017:

13952 Enhanced error detection and reporting for all file I/O
13953 operations.

13955  Documentation:

13957 Programmer Reference updated to version 1.06.



13961 ----------------------------------------
13962 Summary of changes for this label: 04_13_01

13964  ACPI CA Core Subsystem:

13966 Restructured support for BufferFields and RegionFields.
13967 BankFields support is now fully operational.  All known 32-bit
13968 limitations on field sizes have been removed.  Both BufferFields
13969 and (Operation) RegionFields are now supported by the same field
13970 management code.

13972 Resource support now supports QWORD address and IO resources. The
13973 16/32/64 bit address structures and the Extended IRQ structure
13974 have been changed to properly handle Source Resource strings.

13976 A ThreadId of -1 is now used to indicate a "mutex not acquired"
13977 condition internally and must never be returned by AcpiOsThreadId.
13978 This reserved value was changed from 0 since Unix systems allow a
13979 thread ID of 0.

13981 Linux:

13983 Driver code reorganized to enhance portability

13985 Added a kernel configuration option to control ACPI_DEBUG

13987 Fixed the EC driver to honor _GLK.

13989 ASL Compiler, version X2016:

13991 Fixed support for the "FixedHw" keyword.  Previously, the FixedHw
13992 address space was set to 0, not 0x7f as it should be.

13994  ----------------------------------------
13995 Summary of changes for this label: 03_13_01

13997  ACPI CA Core Subsystem:

13999 During ACPI initialization, the _SB_._INI method is now run if
14000 present.

14002 Notify handler fix - notifies are deferred until the parent method
14003 completes execution.  This fixes the "mutex already acquired"
14004 issue seen occasionally.

14006 Part of the "implicit conversion" rules in ACPI 2.0 have been
14007 found to cause compatibility problems with existing ASL/AML.  The
14008 convert "result-to-target-type" implementation has been removed
14009 for stores to method Args and Locals.  Source operand conversion
14010 is still fully implemented.  Possible changes to ACPI 2.0
14011 specification pending.

14013 Fix to AcpiRsCalculatePciRoutingTableLength to return correct
14014 length.
```



```
14016 Fix for compiler warnings for 64-bit compiles.

14018  Linux:

14020 /proc output aligned for easier parsing.

14022 Release-version compile problem fixed.

14024 New kernel configuration options documented in Configure.help.

14026 IBM 600E - Fixed Sleep button may generate "Invalid <NULL>
14027 context" message.

14029  OSPM:

14031 Power resource driver integrated with bus manager.

14033 Fixed kernel fault during active cooling for thermal zones.

14035 Source Code:

14037 The source code tree has been restructured.



14041 ----------------------------------------
14042 Summary of changes for this label: 03_02_01

14044  Linux OS Services Layer (OSL):

14046 Major revision of all Linux-specific code.

14048 Modularized all ACPI-specific drivers.

14050 Added new thermal zone and power resource drivers.

14052 Revamped /proc interface (new functionality is under /proc/acpi).

14054 New kernel configuration options.

14056  Linux known issues:

14058 New kernel configuration options not documented in Configure.help
14059 yet.


14062 Module dependencies not currently implemented. If used, they
14063 should be loaded in this order: busmgr, power, ec, system,
14064 processor, battery, ac_adapter, button, thermal.

14066 Modules will not load if CONFIG_MODVERSION is set.

14068 IBM 600E - entering S5 may reboot instead of shutting down.

14070 IBM 600E - Sleep button may generate "Invalid <NULL> context"
14071 message.

14073 Some systems may fail with "execution mutex already acquired"
14074 message.

14076  ACPI CA Core Subsystem:

14078 Added a new OSL Interface, AcpiOsGetThreadId.  This was required
14079 for the  deadlock detection code. Defined to return a non-zero, 32-
14080 bit thread ID for the currently executing thread.  May be a non-
```

zero constant integer on single-thread systems.

Implemented deadlock detection for internal subsystem mutexes. We
may add conditional compilation for this code (debug only) later.

ASL/AML Mutex object semantics are now fully supported. This
includes multiple acquires/releases by owner and support for the

Mutex SyncLevel parameter.

A new "Force Release" mechanism automatically frees all ASL
Mutexes that have been acquired but not released when a thread
exits the interpreter. This forces conformance to the ACPI spec
("All mutexes must be released when an invocation exits") and
prevents deadlocked ASL threads. This mechanism can be expanded
(later) to monitor other resource acquisitions if OEM ASL code
continues to misbehave (which it will).

Several new ACPI exception codes have been added for the Mutex
support.

Recursive method calls are now allowed and supported (the ACPI
spec does in fact allow recursive method calls.) The number of
recursive calls is subject to the restrictions imposed by the
SERIALIZED method keyword and SyncLevel (ACPI 2.0) method
parameter.

Implemented support for the SyncLevel parameter for control
methods (ACPI 2.0 feature)

Fixed a deadlock problem when multiple threads attempted to use
the interpreter.

Fixed a problem where the string length of a String package
element was not always set in a package returned from
AcpiEvaluateObject.

Fixed a problem where the length of a String package element was
not always included in the length of the overall package returned
from AcpiEvaluateObject.

Added external interfaces (Acpi*) to the ACPI debug memory
manager. This manager keeps a list of all outstanding
allocations, and can therefore detect memory leaks and attempts to
free memory blocks more than once. Useful for code such as the
power manager, etc. May not be appropriate for device drivers.
Performance with the debug code enabled is slow.

The ACPI Global Lock is now an optional hardware element.

ASL Compiler Version X2015:

Integrated changes to allow the compiler to be generated on
multiple platforms.

Linux makefile added to generate the compiler on Linux

Source Code:

All platform-specific headers have been moved to their own
subdirectory, Include/Platform.

New source file added, Interpreter/ammutex.c

New header file, Include/acstruct.h

Documentation:

The programmer reference has been updated for the following new
interfaces: AcpiOsGetThreadId AcpiAllocate AcpiCallocate AcpiFree

---
Summary of changes for this label: 02_08_01

Core ACPI CA Subsystem: Fixed a problem where an error was
incorrectly returned if the return resource buffer was larger than
the actual data (in the resource interfaces).

References to named objects within packages are resolved to the

full pathname string before packages are returned directly (via
the AcpiEvaluateObject interface) or indirectly via the resource
interfaces.

Linux OS Services Layer (OSL):

Improved /proc battery interface.

Added C-state debugging output and other miscellaneous fixes.

ASL Compiler Version X2014:

All defined method arguments can now be used as local variables,
including the ones that are not actually passed in as parameters.
The compiler tracks initialization of the arguments and issues an
exception if they are used without prior assignment (just like
locals).

The -o option now specifies a filename prefix that is used for all
output files, including the AML output file. Otherwise, the
default behavior is as follows: 1) the AML goes to the file
specified in the DSDT. 2) all other output files use the input
source filename as the base.

Summary of changes for this label: 01_25_01

Core ACPI CA Subsystem: Restructured the implementation of object
store support within the interpreter. This includes support for
the Store operator as well as any ASL operators that include a
target operand.

Partially implemented support for Implicit Result-to-Target
conversion. This is when a result object is converted on the fly
to the type of an existing target object. Completion of this
support is pending further analysis of the ACPI specification
concerning this matter.

CPU-specific code has been removed from the subsystem (hardware
directory).

New Power Management Timer functions added

Linux OS Services Layer (OSL): Moved system state transition code
to the core, fixed it, and modified Linux OSL accordingly.

Fixed C2 and C3 latency calculations.

We no longer use the compilation date for the version message on
initialization, but retrieve the version from AcpiGetSystemInfo().

Incorporated for fix Sony VAIO machines.

Documentation: The Programmer Reference has been updated and
reformatted.

ASL Compiler: Version X2013: Fixed a problem where the line
numbering and error reporting could get out of sync in the
presence of multiple include files.

Summary of changes for this label: 01_15_01

Core ACPI CA Subsystem:

Implemented support for type conversions in the execution of the
ASL Concatenate operator (The second operand is converted to
match the type of the first operand before concatenation.)

Support for implicit source operand conversion is partially
implemented.   The ASL source operand types Integer, Buffer, and
String are freely interchangeable for most ASL operators and are
converted by the interpreter on the fly as required. Implicit
Target operand conversion (where the result is converted to the
target type before storing) is not yet implemented.

Support for 32-bit and 64-bit BCD integers is implemented.

Problem fixed where a field read on an aligned field could cause a
read past the end of the field.

New exception, AE_AML_NO_RETURN_VALUE, is returned when a method
does not return a value, but the caller expects one. (The ASL
compiler flags this as a warning.)

ASL Compiler:

Version X2011:
1. Static typechecking of all operands is implemented. This
prevents the use of invalid objects (such as using a Package where
an Integer is required) at compile time instead of at interpreter
run-time.
2. The ASL source line is printed with ALL errors and warnings.
3. Bug fix for source EOF without final linefeed.
4. Debug option is split into a parse trace and a namespace trace.
5. Namespace output option (-n) includes initial values for
integers and strings.
6. Parse-only option added for quick syntax checking.
7. Compiler checks for duplicate ACPI name declarations

Version X2012:
1. Relaxed typechecking to allow interchangeability between
strings, integers, and buffers. These types are now converted by
the interpreter at runtime.
2. Compiler reports time taken by each internal subsystem in the
debug   output file.

Summary of changes for this label: 12_14_00

ASL Compiler:

This is the first official release of the compiler. Since the
compiler requires elements of the Core Subsystem, this label

synchronizes everything.

Summary of changes for this label: 12_08_00

Fixed a problem where named references within the ASL definition
of both OperationRegions and CreateXXXFields did not work
properly. The symptom was an AE_AML_OPERAND_TYPE during
initialization of the region/field. This is similar (but not
related internally) to the problem that was fixed in the last
label.

Implemented both 32-bit and 64-bit support for the BCD ASL
functions ToBCD and FromBCD.

Updated all legal headers to include "2000" in the copyright
years.

Summary of changes for this label: 12_01_00

Fixed a problem where method invocations within the ASL definition
of both OperationRegions and CreateXXXFields did not work
properly. The symptom was an AE_AML_OPERAND_TYPE during
initialization of the region/field:

  nsinit-0209: AE_AML_OPERAND_TYPE while getting region arguments
[DEBG]   ammonad-0284: Exec_monadic2_r/Not: bad operand(s)
(0x3005)

Fixed a problem where operators with more than one nested
subexpression would fail. The symptoms were varied, by mostly
AE_AML_OPERAND_TYPE errors. This was actually a rather serious
problem that has gone unnoticed until now.

  Subtract (Add (1,2), Multiply (3,4))

Fixed a problem where AcpiGetHandle didn’t quite get fixed in the
previous build (The prefix part of a relative path was handled
incorrectly).

Fixed a problem where Operation Region initialization failed if
the operation region name was a "namepath" instead of a simple
"nameseg". Symptom was an AE_NO_OPERAND error.

Fixed a problem where an assignment to a local variable via the
indirect RefOf mechanism only worked for the first such
assignment. Subsequent assignments were ignored.

Summary of changes for this label: 11_15_00

ACPI 2.0 table support with backwards support for ACPI 1.0 and the
0.71 extensions. Note: although we can read ACPI 2.0 BIOS tables,
the AML interpreter does NOT have support for the new 2.0 ASL
grammar terms at this time.

All ACPI hardware access is via the GAS structures in the ACPI 2.0
FADT.

All physical memory addresses across all platforms are now 64 bits
wide. Logical address width remains dependent on the platform
(i.e., "void *").

AcpiOsMapMemory interface changed to a 64-bit physical address.

14346 The AML interpreter integer size is now 64 bits, as per the ACPI
14347 2.0 specification.

14349 For backwards compatibility with ACPI 1.0, ACPI tables with a
14350 revision number less than 2 use 32-bit integers only.

14352 Fixed a problem where the evaluation of OpRegion operands did not
14353 always resolve them to numbers properly.

14355 --
14356 Summary of changes for this label: 10_20_00

14358 Fix for CBN_._STA issue. This fix will allow correct access to
14359 CBN_ OpRegions when the _STA returns 0x8.

14361 Support to convert ACPI constants (Ones, Zeros, One) to actual
14362 values before a package object is returned

14364 Fix for method call as predicate to if/while construct causing
14365 incorrect if/while behavior

14367 Fix for Else block package lengths sometimes calculated wrong (if
14368 block > 63 bytes)

14370 Fix for Processor object length field, was always zero

14372 Table load abort if FACP sanity check fails

14374 Fix for problem with Scope(name) if name already exists

14376 Warning emitted if a named object referenced cannot be found
14377 (resolved) during method execution.

14383 --
14384 Summary of changes for this label: 9_29_00

14386 New table initialization interfaces: AcpiInitializeSubsystem no
14387 longer has any parameters AcpiFindRootPointer - Find the RSDP (if
14388 necessary) AcpiLoadTables (RSDP) - load all tables found at RSDP-
14389 >RSDT Obsolete Interfaces AcpiLoadFirmwareTables - replaced by
14390 AcpiLoadTables

14392 Note: These interface changes require changes to all existing OSDs

14394 The PCI_Config default address space handler is always installed
14395 at the root namespace object.

14397 --
14398 Summary of changes for this label: 09_15_00

14400 The new initialization architecture is implemented. New
14401 interfaces are: AcpiInitializeSubsystem (replaces AcpiInitialize)
14402 AcpiEnableSubsystem Obsolete Interfaces: AcpiLoadNamespace

14404 (Namespace is automatically loaded when a table is loaded)

14406 The ACPI_OPERAND_OBJECT has been optimized to shrink its size from
14407 52 bytes to 32 bytes. There is usually one of these for every
14408 namespace object, so the memory savings is significant.

14410 Implemented just-in-time evaluation of the CreateField operators.

14412 Bug fixes for IA-64 support have been integrated.

14414 Additional code review comments have been implemented

14416 The so-called "third pass parse" has been replaced by a final walk
14417 through the namespace to initialize all operation regions (address
14418 spaces) and fields that have not yet been initialized during the
14419 execution of the various _INI and REG methods.

14421 New file - namespace/nsinit.c

14423 --
14424 Summary of changes for this label: 09_01_00

14426 Namespace manager data structures have been reworked to change the
14427 primary object from a table to a single object. This has
14428 resulted in dynamic memory savings of 3X within the namespace and
14429 2X overall in the ACPI CA subsystem.

14431 Fixed problem where the call to AcpiEvFindPciRootBuses was
14432 inadvertently left commented out.

14434 Reduced the warning count when generating the source with the GCC
14435 compiler.

14437 Revision numbers added to each module header showing the
14438 SourceSafe version of the file. Please refer to this version
14439 number when giving us feedback or comments on individual modules.

14441 The main object types within the subsystem have been renamed to
14442 clarify their purpose:

14444 ACPI_INTERNAL_OBJECT -> ACPI_OPERAND_OBJECT
14445 ACPI_GENERIC_OP -> ACPI_PARSE_OBJECT
14446 ACPI_NAME_TABLE_ENTRY -> ACPI_NAMESPACE_NODE

14448 NOTE: no changes to the initialization sequence are included in
14449 this label.

14451 --
14452 Summary of changes for this label: 08_23_00

14454 Fixed problem where TerminateControlMethod was being called
14455 multiple times per method

14457 Fixed debugger problem where single stepping caused a semaphore to
14458 be oversignalled

14460 Improved performance through additional parse object caching -
14461 added ACPI_EXTENDED_OP type

14463 --
14464 Summary of changes for this label: 08_10_00

14466 Parser/Interpreter integration: Eliminated the creation of
14467 complete parse trees for ACPI tables and control methods.
14468 Instead, parse subtrees are created and then deleted as soon as
14469 they are processed (Either entered into the namespace or executed
14470 by the interpreter). This reduces the use of dynamic kernel
14471 memory significantly. (about 10X)

14473 Exception codes broken into classes and renumbered. Be sure to
14474 recompile all code that includes acexcep.h. Hopefully we won't
14475 have to renumber the codes again now that they are split into
14476 classes (environment, programmer, AML code, ACPI table, and

internal).

Fixed some additional alignment issues in the Resource Manager
subcomponent

Implemented semaphore tracking in the AcpiExec utility, and fixed
several places where mutexes/semaphores were being unlocked
without a corresponding lock operation. There are no known
semaphore or mutex "leaks" at this time.

Fixed the case where an ASL Return operator is used to return an
unnamed package.

Summary of changes for this label: 07_28_00

Fixed a problem with the way addresses were calculated in
AcpiAmlReadFieldData() and AcpiAmlWriteFieldData(). This problem
manifested itself when a Field was created with WordAccess or
DwordAccess, but the field unit defined within the Field was less

than a Word or Dword.

Fixed a problem in AmlDumpOperands() module's loop to pull
operands off of the operand stack to display information. The
problem manifested itself as a TLB error on 64-bit systems when
accessing an operand stack with two or more operands.

Fixed a problem with the PCI configuration space handlers where
context was getting confused between accesses. This required a
change to the generic address space handler and address space
setup definitions. Handlers now get both a global handler context
(this is the one passed in by the user when executing
AcpiInstallAddressSpaceHandler() and a specific region context
that is unique to each region (For example, the _ADR, _SEG and
_BBN values associated with a specific region). The generic
function definitions have changed to the following:

typedef ACPI_STATUS (*ADDRESS_SPACE_HANDLER) (UINT32 Function,
UINT32 Address, UINT32 BitWidth, UINT32 *Value, void
*HandlerContext, // This used to be void *Context void
*RegionContext); // This is an additional parameter

typedef ACPI_STATUS (*ADDRESS_SPACE_SETUP) (ACPI_HANDLE
RegionHandle, UINT32 Function, void *HandlerContext, void
**RegionContext); // This used to be **ReturnContext

Summary of changes for this label: 07_21_00

Major file consolidation and rename. All files within the
interpreter have been renamed as well as most header files. This
was done to prevent collisions with existing files in the host
OSs -- filenames such as "config.h" and "global.h" seem to be
quite common. The VC project files have been updated. All
makefiles will require modification.

The parser/interpreter integration continues in Phase 5 with the
implementation of a complete 2-pass parse (the AML is parsed
twice) for each table; This avoids the construction of a huge
parse tree and therefore reduces the amount of dynamic memory
required by the subsystem. Greater use of the parse object cache
means that performance is unaffected.

Many comments from the two code reviews have been rolled in.

The 64-bit alignment support is complete.

Summary of changes for this label: 06_30_00

With a nod and a tip of the hat to the technology of yesteryear,
we've added support in the source code for 80 column output
devices. The code is now mostly constrained to 80 columns or
less to support environments and editors that 1) cannot display
or print more than 80 characters on a single line, and 2) cannot
disable line wrapping.

A major restructuring of the namespace data structure has been
completed. The result is 1) cleaner and more
understandable/maintainable code, and 2) a significant reduction
in the dynamic memory requirement for each named ACPI object
(almost half).

Summary of changes for this label: 06_23_00

Linux support has been added. In order to obtain approval to get
the ACPI CA subsystem into the Linux kernel, we've had to make
quite a few changes to the base subsystem that will affect all
users (all the changes are generic and OS- independent). The
effects of these global changes have been somewhat far reaching.
Files have been merged and/or renamed and interfaces have been
renamed. The major changes are described below.

Osd* interfaces renamed to AcpiOs* to eliminate namespace
pollution/confusion within our target kernels. All OSD
interfaces must be modified to match the new naming convention.

Files merged across the subsystem. A number of the smaller source
and header files have been merged to reduce the file count and
increase the density of the existing files. There are too many
to list here. In general, makefiles that call out individual
files will require rebuilding.

Interpreter files renamed. All interpreter files now have the
prefix am* instead of ie* and is*.

Header files renamed: The acapi.h file is now acpixf.h. The
acpiosd.h file is now acpiosxf.h. We are removing references to
the acronym "API" since it is somewhat windowsy. The new name is
"external interface" or xface or xf in the filenames.j

All manifest constants have been forced to upper case (some were
mixed case.) Also, the string "ACPI_" has been prepended to many
(not all) of the constants, typedefs, and structs.

The globals "DebugLevel" and "DebugLayer" have been renamed
"AcpiDbgLevel" and "AcpiDbgLayer" respectively.

All other globals within the subsystem are now prefixed with
"AcpiGbl_" Internal procedures within the subsystem are now
prefixed with "Acpi" (with only a few exceptions). The original
two-letter abbreviation for the subcomponent remains after "Acpi"
- for example, CmCallocate became AcpiCmCallocate.

Added a source code translation/conversion utility. Used to
generate the Linux source code, it can be modified to generate
other types of source as well. Can also be used to cleanup
existing source by removing extraneous spaces and blank lines.
Found in tools/acpisrc/*

OsdUnMapMemory was renamed to OsdUnmapMemory and then
AcpiOsUnmapMemory. (UnMap became Unmap).

A "MaxUnits" parameter has been added to AcpiOsCreateSemaphore.
When set to one, this indicates that the caller wants to use the

semaphore as a mutex, not a counting semaphore. ACPI CA uses
both types. However, implementers of this call may want to use
different OS primitives depending on the type of semaphore
requested. For example, some operating systems provide separate

"mutex" and "semaphore" interfaces - where the mutex interface is
much faster because it doesn't have all the overhead of a full
semaphore implementation.

Fixed a deadlock problem where a method that accesses the PCI
address space can block forever if it is the first access to the
space.

Summary of changes for this label: 06_02_00

Support for environments that cannot handle unaligned data
accesses (e.g. firmware and OS environments devoid of alignment
handler technology namely SAL/EFI and the IA-64 Linux kernel) has
been added (via configurable macros) in these three areas: -
Transfer of data from the raw AML byte stream is done via byte
moves instead of word/dword/qword moves. - External objects are
aligned within the user buffer, including package elements (sub-
objects). - Conversion of name strings to UINT32 Acpi Names is now
done byte-wise.

The Store operator was modified to mimic Microsoft's
implementation when storing to a Buffer Field.

Added a check of the BM_STS bit before entering C3.

The methods subdirectory has been obsoleted and removed. A new
file, cmeval.c subsumes the functionality.

A 16-bit (DOS) version of AcpiExec has been developed. The
makefile is under the acpiexec directory.

```
**********************************************************
   13858 Thu Dec 26 13:48:22 2013
new/usr/src/common/acpica/common/acgetline.c
update to acpica-unix2-20130927
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acgetline - local line editing
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acparser.h"
#include "acdebug.h"

#include <stdio.h>

/*
 * This is an os-independent implementation of line-editing services needed
 * by the AcpiExec utility. It uses getchar() and putchar() and the existing
 * history support provided by the AML debugger. It assumes that the terminal
 * is in the correct line-editing mode such as raw and noecho. The OSL
 * interface AcpiOsInitialize should do this. AcpiOsTerminate should put the
 * terminal back into the original mode.
 */
#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("acgetline")
```



```
/* Local prototypes */

static void
AcpiAcClearLine (
    UINT32                  EndOfLine,
    UINT32                  CursorPosition);

/* Various ASCII constants */

#define _ASCII_NUL                  0
#define _ASCII_BACKSPACE            0x08
#define _ASCII_TAB                  0x09
#define _ASCII_ESCAPE               0x1B
#define _ASCII_SPACE                0x20
#define _ASCII_LEFT_BRACKET         0x5B
#define _ASCII_DEL                  0x7F
#define _ASCII_UP_ARROW             'A'
#define _ASCII_DOWN_ARROW           'B'
#define _ASCII_RIGHT_ARROW          'C'
#define _ASCII_LEFT_ARROW           'D'
#define _ASCII_NEWLINE              '\n'

extern UINT32               AcpiGbl_NextCmdNum;

/* Erase a single character on the input command line */

#define ACPI_CLEAR_CHAR() \
    putchar (_ASCII_BACKSPACE); \
    putchar (_ASCII_SPACE); \
    putchar (_ASCII_BACKSPACE);

/* Backup cursor by Count positions */

#define ACPI_BACKUP_CURSOR(i, Count) \
    for (i = 0; i < (Count); i++) \
        {putchar (_ASCII_BACKSPACE);}


/******************************************************************************
 *
 * FUNCTION:    AcpiAcClearLine
 *
 * PARAMETERS:  EndOfLine           - Current end-of-line index
 *              CursorPosition      - Current cursor position within line
 *
 * RETURN:      None
 *
 * DESCRIPTION: Clear the entire command line the hard way, but probably the
 *              most portable.
 *
 *****************************************************************************/

static void
AcpiAcClearLine (
    UINT32                  EndOfLine,
    UINT32                  CursorPosition)
{
    UINT32                  i;


    if (CursorPosition < EndOfLine)
    {
        /* Clear line from current position to end of line */
```

```
128         for (i = 0; i < (EndOfLine - CursorPosition); i++)
129         {
130             putchar (' ');
131         }
132     }

134     /* Clear the entire line */

136     for (; EndOfLine > 0; EndOfLine--)
137     {
138         ACPI_CLEAR_CHAR ();
139     }
140 }


143 /******************************************************************************
144  *
145  * FUNCTION:    AcpiOsGetLine
146  *
147  * PARAMETERS:  Buffer              - Where to return the command line
148  *              BufferLength        - Maximum length of Buffer
149  *              BytesRead           - Where the actual byte count is returned
150  *
151  * RETURN:      Status and actual bytes read
152  *
153  * DESCRIPTION: Get the next input line from the terminal. NOTE: terminal
154  *              is expected to be in a mode that supports line-editing (raw,
155  *              noecho). This function is intended to be very portable. Also,
156  *              it uses the history support implemented in the AML debugger.
157  *
158  *****************************************************************************/

160 ACPI_STATUS
161 AcpiOsGetLine (
162     char                    *Buffer,
163     UINT32                  BufferLength,
164     UINT32                  *BytesRead)
165 {
166     char                    *NextCommand;
167     UINT32                  MaxCommandIndex = AcpiGbl_NextCmdNum - 1;
168     UINT32                  CurrentCommandIndex = MaxCommandIndex;
169     UINT32                  PreviousCommandIndex = MaxCommandIndex;
170     int                     InputChar;
171     UINT32                  CursorPosition = 0;
172     UINT32                  EndOfLine = 0;
173     UINT32                  i;


176     /* Always clear the line buffer before we read a new line */

178     memset (Buffer, 0, BufferLength);

180     /*
181      * This loop gets one character at a time (except for esc sequences)
182      * until a newline or error is detected.
183      *
184      * Note: Don't attempt to write terminal control ESC sequences, even
185      * though it makes certain things more difficult.
186      */
187     while (1)
188     {
189         if (EndOfLine >= (BufferLength - 1))
190         {
191             return (AE_BUFFER_OVERFLOW);
192         }
```



```
194         InputChar = getchar ();
195         switch (InputChar)
196         {
197         default: /* This is the normal character case */

199             /* Echo the character (at EOL) and copy it to the line buffer */

201             if (EndOfLine == CursorPosition)
202             {
203                 putchar (InputChar);
204                 Buffer[EndOfLine] = (char) InputChar;

206                 EndOfLine++;
207                 CursorPosition++;
208                 Buffer[EndOfLine] = 0;
209                 continue;
210             }

212             /* Insert character into the middle of the buffer */

214             memmove (&Buffer[CursorPosition + 1], &Buffer[CursorPosition],
215                 (EndOfLine - CursorPosition + 1));

217             Buffer [CursorPosition] = (char) InputChar;
218             Buffer [EndOfLine + 1] = 0;

220             /* Display the new part of line starting at the new character */

222             fprintf (stdout, "%s", &Buffer[CursorPosition]);

224             /* Restore cursor */

226             ACPI_BACKUP_CURSOR (i, EndOfLine - CursorPosition);
227             CursorPosition++;
228             EndOfLine++;
229             continue;

231         case _ASCII_DEL: /* Backspace key */

233             if (!EndOfLine) /* Any characters on the command line? */
234             {
235                 continue;
236             }

238             if (EndOfLine == CursorPosition) /* Erase the final character */
239             {
240                 ACPI_CLEAR_CHAR ();
241                 EndOfLine--;
242                 CursorPosition--;
243                 continue;
244             }

246             if (!CursorPosition) /* Do not backup beyond start of line */
247             {
248                 continue;
249             }

251             /* Remove the character from the line */

253             memmove (&Buffer[CursorPosition - 1], &Buffer[CursorPosition],
254                 (EndOfLine - CursorPosition + 1));

256             /* Display the new part of line starting at the new character */

258             putchar (_ASCII_BACKSPACE);
259             fprintf (stdout, "%s ", &Buffer[CursorPosition - 1]);
```

```
261             /* Restore cursor */

263             ACPI_BACKUP_CURSOR (i, EndOfLine - CursorPosition + 1);
264             EndOfLine--;
265             if (CursorPosition > 0)
266             {
267                 CursorPosition--;
268             }
269             continue;

271         case _ASCII_NEWLINE: /* Normal exit case at end of command line */
272         case _ASCII_NUL:

274             /* Return the number of bytes in the command line string */

276             if (BytesRead)
277             {
278                 *BytesRead = EndOfLine;
279             }

281             /* Echo, terminate string buffer, and exit */

283             putchar (InputChar);
284             Buffer[EndOfLine] = 0;
285             return (AE_OK);

287         case _ASCII_TAB:

289             /* Ignore */

291             continue;

293         case EOF:

295             return (AE_ERROR);

297         case _ASCII_ESCAPE:

299             /* Check for escape sequences of the form "ESC[x" */

301             InputChar = getchar ();
302             if (InputChar != _ASCII_LEFT_BRACKET)
303             {
304                 continue; /* Ignore this ESC, does not have the '[' */
305             }

307             /* Get the code following the ESC [ */

309             InputChar = getchar (); /* Backup one character */
310             switch (InputChar)
311             {
312             case _ASCII_LEFT_ARROW:

314                 if (CursorPosition > 0)
315                 {
316                     putchar (_ASCII_BACKSPACE);
317                     CursorPosition--;
318                 }
319                 continue;

321             case _ASCII_RIGHT_ARROW:
322                 /*
323                  * Move one character forward. Do this without sending
324                  * ESC sequence to the terminal for max portability.
325                  */
```



```
326                 if (CursorPosition < EndOfLine)
327                 {
328                     /* Backup to start of line and print the entire line */

330                     ACPI_BACKUP_CURSOR (i, CursorPosition);
331                     fprintf (stdout, "%s", Buffer);

333                     /* Backup to where the cursor should be */

335                     CursorPosition++;
336                     ACPI_BACKUP_CURSOR (i, EndOfLine - CursorPosition);
337                 }
338                 continue;

340             case _ASCII_UP_ARROW:

342                 /* If no commands available or at start of history list, ignore

344                 if (!CurrentCommandIndex)
345                 {
346                     continue;
347                 }

349                 /* Manage our up/down progress */

351                 if (CurrentCommandIndex > PreviousCommandIndex)
352                 {
353                     CurrentCommandIndex = PreviousCommandIndex;
354                 }

356                 /* Get the historical command from the debugger */

358                 NextCommand = AcpiDbGetHistoryByIndex (CurrentCommandIndex);
359                 if (!NextCommand)
360                 {
361                     return (AE_ERROR);
362                 }

364                 /* Make this the active command and echo it */

366                 AcpiAcClearLine (EndOfLine, CursorPosition);
367                 strcpy (Buffer, NextCommand);
368                 fprintf (stdout, "%s", Buffer);
369                 EndOfLine = CursorPosition = strlen (Buffer);

371                 PreviousCommandIndex = CurrentCommandIndex;
372                 CurrentCommandIndex--;
373                 continue;

375             case _ASCII_DOWN_ARROW:

377                 if (!MaxCommandIndex) /* Any commands available? */
378                 {
379                     continue;
380                 }

382                 /* Manage our up/down progress */

384                 if (CurrentCommandIndex < PreviousCommandIndex)
385                 {
386                     CurrentCommandIndex = PreviousCommandIndex;
387                 }

389                 /* If we are the end of the history list, output a clear new lin

391                 if ((CurrentCommandIndex + 1) > MaxCommandIndex)
```

```
392                 {
393                     AcpiAcClearLine (EndOfLine, CursorPosition);
394                     EndOfLine = CursorPosition = 0;
395                     PreviousCommandIndex = CurrentCommandIndex;
396                     continue;
397                 }

399                 PreviousCommandIndex = CurrentCommandIndex;
400                 CurrentCommandIndex++;

402                  /* Get the historical command from the debugger */

404                 NextCommand = AcpiDbGetHistoryByIndex (CurrentCommandIndex);
405                 if (!NextCommand)
406                 {
407                     return (AE_ERROR);
408                 }

410                 /* Make this the active command and echo it */

412                 AcpiAcClearLine (EndOfLine, CursorPosition);
413                 strcpy (Buffer, NextCommand);
414                 fprintf (stdout, "%s", Buffer);
415                 EndOfLine = CursorPosition = strlen (Buffer);
416                 continue;

418             case 0x31:
419             case 0x32:
420             case 0x33:
421             case 0x34:
422             case 0x35:
423             case 0x36:
424                 /*
425                  * Ignore the various keys like insert/delete/home/end, etc.
426                  * But we must eat the final character of the ESC sequence.
427                  */
428                 InputChar = getchar ();
429                 continue;

431             default:

433                 /* Ignore random escape sequences that we don't care about */

435                 continue;
436             }
437             continue;
438         }
439     }
440 }
```

```
**********************************************************
    9651 Thu Dec 26 13:48:22 2013
new/usr/src/common/acpica/common/adfile.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: adfile - Application-level disassembler file support routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acapps.h"

#include <stdio.h>


#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("adfile")

/* Local prototypes */

static INT32
```



```
AdWriteBuffer (
    char                    *Filename,
    char                    *Buffer,
    UINT32                  Length);

static char                 FilenameBuf[20];


/******************************************************************************
 *
 * FUNCTION:    AfGenerateFilename
 *
 * PARAMETERS:  Prefix              - prefix string
 *              TableId             - The table ID
 *
 * RETURN:      Pointer to the completed string
 *
 * DESCRIPTION: Build an output filename from an ACPI table ID string
 *
 *****************************************************************************/

char *
AdGenerateFilename (
    char                    *Prefix,
    char                    *TableId)
{
    UINT32                  i;
    UINT32                  j;


    for (i = 0; Prefix[i]; i++)
    {
        FilenameBuf[i] = Prefix[i];
    }

    FilenameBuf[i] = '_';
    i++;

    for (j = 0; j < 8 && (TableId[j] != ' ') && (TableId[j] != 0); i++, j++)
    {
        FilenameBuf[i] = TableId[j];
    }

    FilenameBuf[i] = 0;
    strcat (FilenameBuf, ACPI_TABLE_FILE_SUFFIX);
    return (FilenameBuf);
}


/******************************************************************************
 *
 * FUNCTION:    AfWriteBuffer
 *
 * PARAMETERS:  Filename            - name of file
 *              Buffer              - data to write
 *              Length              - length of data
 *
 * RETURN:      Actual number of bytes written
 *
 * DESCRIPTION: Open a file and write out a single buffer
 *
 *****************************************************************************/

static INT32
AdWriteBuffer (
    char                    *Filename,
```

```
124     char                    *Buffer,
125     UINT32                  Length)
126 {
127     FILE                    *File;
128     ACPI_SIZE               Actual;


131     File = fopen (Filename, "wb");
132     if (!File)
133     {
134         printf ("Could not open file %s\n", Filename);
135         return (-1);
136     }

138     Actual = fwrite (Buffer, 1, (size_t) Length, File);
139     if (Actual != Length)
140     {
141         printf ("Could not write to file %s\n", Filename);
142     }

144     fclose (File);
145     return ((INT32) Actual);
146 }


149 /******************************************************************************
150  *
151  * FUNCTION:    AfWriteTable
152  *
153  * PARAMETERS:  Table               - pointer to the ACPI table
154  *              Length              - length of the table
155  *              TableName           - the table signature
156  *              OemTableID          - from the table header
157  *
158  * RETURN:      None
159  *
160  * DESCRIPTION: Dump the loaded tables to a file (or files)
161  *
162  *****************************************************************************/

164 void
165 AdWriteTable (
166     ACPI_TABLE_HEADER       *Table,
167     UINT32                  Length,
168     char                    *TableName,
169     char                    *OemTableId)
170 {
171     char                    *Filename;


174     Filename = AdGenerateFilename (TableName, OemTableId);
175     AdWriteBuffer (Filename, (char *) Table, Length);

177     AcpiOsPrintf ("Table [%s] written to \"%s\"\n", TableName, Filename);
178 }


181 /******************************************************************************
182  *
183  * FUNCTION:    FlGenerateFilename
184  *
185  * PARAMETERS:  InputFilename       - Original ASL source filename
186  *              Suffix              - New extension.
187  *
188  * RETURN:      New filename containing the original base + the new suffix
189  *
```



```
190  * DESCRIPTION: Generate a new filename from the ASL source filename and a new
191  *              extension. Used to create the *.LST, *.TXT, etc. files.
192  *
193  *****************************************************************************/

195 char *
196 FlGenerateFilename (
197     char                    *InputFilename,
198     char                    *Suffix)
199 {
200     char                    *Position;
201     char                    *NewFilename;
202     char                    *DirectoryPosition;


205     /*
206      * Copy the original filename to a new buffer. Leave room for the worst case
207      * where we append the suffix, an added dot and the null terminator.
208      */
209     NewFilename = ACPI_ALLOCATE_ZEROED ((ACPI_SIZE)
210         strlen (InputFilename) + strlen (Suffix) + 2);
211     strcpy (NewFilename, InputFilename);

213     /* Try to find the last dot in the filename */

215     DirectoryPosition = strrchr (NewFilename, '/');
216     Position = strrchr (NewFilename, '.');

218     if (Position && (Position > DirectoryPosition))
219     {
220         /* Tack on the new suffix */

222         Position++;
223         *Position = 0;
224         strcat (Position, Suffix);
225     }
226     else
227     {
228         /* No dot, add one and then the suffix */

230         strcat (NewFilename, ".");
231         strcat (NewFilename, Suffix);
232     }

234     return (NewFilename);
235 }


238 /******************************************************************************
239  *
240  * FUNCTION:    FlStrdup
241  *
242  * DESCRIPTION: Local strdup function
243  *
244  *****************************************************************************/

246 static char *
247 FlStrdup (
248     char                *String)
249 {
250     char                *NewString;


253     NewString = ACPI_ALLOCATE ((ACPI_SIZE) strlen (String) + 1);
254     if (!NewString)
255     {
```

```
256         return (NULL);
257     }

259     strcpy (NewString, String);
260     return (NewString);
261 }


264 /******************************************************************************
265  *
266  * FUNCTION:    FlSplitInputPathname
267  *
268  * PARAMETERS:  InputFilename       - The user-specified ASL source file to be
269  *                                    compiled
270  *              OutDirectoryPath    - Where the directory path prefix is
271  *                                    returned
272  *              OutFilename         - Where the filename part is returned
273  *
274  * RETURN:      Status
275  *
276  * DESCRIPTION: Split the input path into a directory and filename part
277  *              1) Directory part used to open include files
278  *              2) Filename part used to generate output filenames
279  *
280  *****************************************************************************/

282 ACPI_STATUS
283 FlSplitInputPathname (
284     char                    *InputPath,
285     char                    **OutDirectoryPath,
286     char                    **OutFilename)
287 {
288     char                    *Substring;
289     char                    *DirectoryPath;
290     char                    *Filename;


293     *OutDirectoryPath = NULL;

295     if (!InputPath)
296     {
297         return (AE_OK);
298     }

300     /* Get the path to the input filename's directory */

302     DirectoryPath = FlStrdup (InputPath);
303     if (!DirectoryPath)
304     {
305         return (AE_NO_MEMORY);
306     }

308     /* Convert backslashes to slashes in the entire path */

310     UtConvertBackslashes (DirectoryPath);

312     /* Backup to last slash or colon */

314     Substring = strrchr (DirectoryPath, '/');
315     if (!Substring)
316     {
317         Substring = strrchr (DirectoryPath, ':');
318     }

320     /* Extract the simple filename */
```



```
322     if (!Substring)
323     {
324         Filename = FlStrdup (DirectoryPath);
325         DirectoryPath[0] = 0;
326     }
327     else
328     {
329         Filename = FlStrdup (Substring + 1);
330         *(Substring+1) = 0;
331     }

333     if (!Filename)
334     {
335         ACPI_FREE (DirectoryPath);
336         return (AE_NO_MEMORY);
337     }

339     *OutDirectoryPath = DirectoryPath;

341     if (OutFilename)
342     {
343         *OutFilename = Filename;
344         return (AE_OK);
345     }

347     ACPI_FREE (Filename);
348     return (AE_OK);
349 }
```

```
**********************************************************
   27624 Thu Dec 26 13:48:23 2013
new/usr/src/common/acpica/common/adisasm.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: adisasm - Application-level disassembler routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdebug.h"
#include "acdisasm.h"
#include "acdispat.h"
#include "acnamesp.h"
#include "actables.h"
#include "acapps.h"

#include <stdio.h>
#include <time.h>


#define _COMPONENT          ACPI_TOOLS
```



```
        ACPI_MODULE_NAME    ("adisasm")

/*
 * Older versions of Bison won't emit this external in the generated header.
 * Newer versions do emit the external, so we don't need to do it.
 */
#ifndef ASLCOMPILER_ASLCOMPILERPARSE_H
extern int                  AslCompilerdebug;
#endif

ACPI_STATUS
NsDisplayNamespace (
    void);

void
NsSetupNamespaceListing (
    void                    *Handle);


/* Local prototypes */

static UINT32
AdGetFileSize (
    FILE                    *File);

static void
AdCreateTableHeader (
    char                    *Filename,
    ACPI_TABLE_HEADER       *Table);

/* Stubs for ASL compiler */

#ifndef ACPI_ASL_COMPILER
BOOLEAN
AcpiDsIsResultUsed (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState)
{
    return TRUE;
}

ACPI_STATUS
AcpiDsMethodError (
    ACPI_STATUS             Status,
    ACPI_WALK_STATE         *WalkState)
{
    return (Status);
}
#endif

ACPI_STATUS
AcpiNsLoadTable (
    UINT32                  TableIndex,
    ACPI_NAMESPACE_NODE     *Node)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiDsRestartControlMethod (
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     *ReturnDesc)
{
    return (AE_OK);
}
```

```
127 void
128 AcpiDsTerminateControlMethod (
129     ACPI_OPERAND_OBJECT     *MethodDesc,
130     ACPI_WALK_STATE         *WalkState)
131 {
132     return;
133 }

135 ACPI_STATUS
136 AcpiDsCallControlMethod (
137     ACPI_THREAD_STATE       *Thread,
138     ACPI_WALK_STATE         *WalkState,
139     ACPI_PARSE_OBJECT       *Op)
140 {
141     return (AE_OK);
142 }

144 ACPI_STATUS
145 AcpiDsMethodDataInitArgs (
146     ACPI_OPERAND_OBJECT     **Params,
147     UINT32                  MaxParamCount,
148     ACPI_WALK_STATE         *WalkState)
149 {
150     return (AE_OK);
151 }


154 static ACPI_TABLE_DESC      LocalTables[1];
155 static ACPI_PARSE_OBJECT    *AcpiGbl_ParseOpRoot;


158 /******************************************************************************
159  *
160  * FUNCTION:    AdGetFileSize
161  *
162  * PARAMETERS:  File                - Open file handle
163  *
164  * RETURN:      File Size
165  *
166  * DESCRIPTION: Get current file size. Uses seek-to-EOF. File must be open.
167  *
168  *****************************************************************************/

170 static UINT32
171 AdGetFileSize (
172     FILE                    *File)
173 {
174     UINT32                  FileSize;
175     long                    Offset;


178     Offset = ftell (File);

180     fseek (File, 0, SEEK_END);
181     FileSize = (UINT32) ftell (File);

183     /* Restore file pointer */

185     fseek (File, Offset, SEEK_SET);
186     return (FileSize);
187 }


190 /******************************************************************************
191  *
192  * FUNCTION:    AdInitialize
```



```
193  *
194  * PARAMETERS:  None
195  *
196  * RETURN:      Status
197  *
198  * DESCRIPTION: ACPICA and local initialization
199  *
200  *****************************************************************************/

202 ACPI_STATUS
203 AdInitialize (
204     void)
205 {
206     ACPI_STATUS             Status;


209     /* ACPI CA subsystem initialization */

211     Status = AcpiOsInitialize ();
212     if (ACPI_FAILURE (Status))
213     {
214         return (Status);
215     }

217     Status = AcpiUtInitGlobals ();
218     if (ACPI_FAILURE (Status))
219     {
220         return (Status);
221     }

223     Status = AcpiUtMutexInitialize ();
224     if (ACPI_FAILURE (Status))
225     {
226         return (Status);
227     }

229     Status = AcpiNsRootInitialize ();
230     if (ACPI_FAILURE (Status))
231     {
232         return (Status);
233     }

235     /* Setup the Table Manager (cheat - there is no RSDT) */

237     AcpiGbl_RootTableList.MaxTableCount = 1;
238     AcpiGbl_RootTableList.CurrentTableCount = 0;
239     AcpiGbl_RootTableList.Tables = LocalTables;

241     return (Status);
242 }


245 /******************************************************************************
246  *
247  * FUNCTION:    AdAmlDisassemble
248  *
249  * PARAMETERS:  Filename            - AML input filename
250  *              OutToFile           - TRUE if output should go to a file
251  *              Prefix              - Path prefix for output
252  *              OutFilename         - where the filename is returned
253  *              GetAllTables        - TRUE if all tables are desired
254  *
255  * RETURN:      Status
256  *
257  * DESCRIPTION: Disassemble an entire ACPI table
258  *
```

```
259  *****************************************************************************/

261 ACPI_STATUS
262 AdAmlDisassemble (
263     BOOLEAN                 OutToFile,
264     char                    *Filename,
265     char                    *Prefix,
266     char                    **OutFilename,
267     BOOLEAN                 GetAllTables)
268 {
269     ACPI_STATUS             Status;
270     char                    *DisasmFilename = NULL;
271     char                    *ExternalFilename;
272     ACPI_EXTERNAL_FILE      *ExternalFileList = AcpiGbl_ExternalFileList;
273     FILE                    *File = NULL;
274     ACPI_TABLE_HEADER       *Table = NULL;
275     ACPI_TABLE_HEADER       *ExternalTable;
276     ACPI_OWNER_ID           OwnerId;


279     /*
280      * Input: AML code from either a file or via GetTables (memory or
281      * registry)
282      */
283     if (Filename)
284     {
285         Status = AcpiDbGetTableFromFile (Filename, &Table);
286         if (ACPI_FAILURE (Status))
287         {
288             return (Status);
289         }

291         /*
292          * External filenames separated by commas
293          * Example: iasl -e file1,file2,file3 -d xxx.aml
294          */
295         while (ExternalFileList)
296         {
297             ExternalFilename = ExternalFileList->Path;
298             if (!ACPI_STRCMP (ExternalFilename, Filename))
299             {
300                 /* Next external file */

302                 ExternalFileList = ExternalFileList->Next;
303                 continue;
304             }

306             Status = AcpiDbGetTableFromFile (ExternalFilename, &ExternalTable);
307             if (ACPI_FAILURE (Status))
308             {
309                 return (Status);
310             }

312             /* Load external table for symbol resolution */

314             if (ExternalTable)
315             {
316                 Status = AdParseTable (ExternalTable, &OwnerId, TRUE, TRUE);
317                 if (ACPI_FAILURE (Status))
318                 {
319                     AcpiOsPrintf ("Could not parse external ACPI tables, %s\n",
320                         AcpiFormatException (Status));
321                     return (Status);
322                 }

324                 /*
```



```
325                  * Load namespace from names created within control methods
326                  * Set owner id of nodes in external table
327                  */
328                 AcpiDmFinishNamespaceLoad (AcpiGbl_ParseOpRoot,
329                     AcpiGbl_RootNode, OwnerId);
330                 AcpiPsDeleteParseTree (AcpiGbl_ParseOpRoot);
331             }

333             /* Next external file */

335             ExternalFileList = ExternalFileList->Next;
336         }

338         /* Clear external list generated by Scope in external tables */

340         if (AcpiGbl_ExternalFileList)
341         {
342             AcpiDmClearExternalList ();
343         }

345         /* Load any externals defined in the optional external ref file */

347         AcpiDmGetExternalsFromFile ();
348     }
349     else
350     {
351         Status = AdGetLocalTables (Filename, GetAllTables);
352         if (ACPI_FAILURE (Status))
353         {
354             AcpiOsPrintf ("Could not get ACPI tables, %s\n",
355                 AcpiFormatException (Status));
356             return (Status);
357         }

359         if (!AcpiGbl_DbOpt_disasm)
360         {
361             return (AE_OK);
362         }

364         /* Obtained the local tables, just disassemble the DSDT */

366         Status = AcpiGetTable (ACPI_SIG_DSDT, 0, &Table);
367         if (ACPI_FAILURE (Status))
368         {
369             AcpiOsPrintf ("Could not get DSDT, %s\n",
370                 AcpiFormatException (Status));
371             return (Status);
372         }

374         AcpiOsPrintf ("\nDisassembly of DSDT\n");
375         Prefix = AdGenerateFilename ("dsdt", Table->OemTableId);
376     }

378     /*
379      * Output: ASL code. Redirect to a file if requested
380      */
381     if (OutToFile)
382     {
383         /* Create/Open a disassembly output file */

385         DisasmFilename = FlGenerateFilename (Prefix, FILE_SUFFIX_DISASSEMBLY);
386         if (!OutFilename)
387         {
388             fprintf (stderr, "Could not generate output filename\n");
389             Status = AE_ERROR;
390             goto Cleanup;
```

```
391         }

393         File = fopen (DisasmFilename, "w+");
394         if (!File)
395         {
396             fprintf (stderr, "Could not open output file %s\n", DisasmFilename);
397             Status = AE_ERROR;
398             goto Cleanup;
399         }

401         AcpiOsRedirectOutput (File);
402     }

404     *OutFilename = DisasmFilename;

406     if (!AcpiUtIsAmlTable (Table))
407     {
408         AdDisassemblerHeader (Filename);
409         AcpiOsPrintf (" * ACPI Data Table [%4.4s]\n *\n",
410             Table->Signature);
411         AcpiOsPrintf (" * Format: [HexOffset DecimalOffset ByteLength]  "
412             "FieldName : FieldValue\n */\n\n");

414         AcpiDmDumpDataTable (Table);
415         fprintf (stderr, "Acpi Data Table [%4.4s] decoded\n",
416             Table->Signature);
417         fprintf (stderr, "Formatted output:  %s - %u bytes\n",
418             DisasmFilename, AdGetFileSize (File));
419     }
420     else
421     {
422         /* Always parse the tables, only option is what to display */

424         Status = AdParseTable (Table, &OwnerId, TRUE, FALSE);
425         if (ACPI_FAILURE (Status))
426         {
427             AcpiOsPrintf ("Could not parse ACPI tables, %s\n",
428                 AcpiFormatException (Status));
429             goto Cleanup;
430         }

432         if (AslCompilerdebug)
433         {
434             AcpiOsPrintf ("/**** Before second load\n");

436             NsSetupNamespaceListing (File);
437             NsDisplayNamespace ();
438             AcpiOsPrintf ("*****/\n");
439         }

441         /* Load namespace from names created within control methods */

443         AcpiDmFinishNamespaceLoad (AcpiGbl_ParseOpRoot,
444             AcpiGbl_RootNode, OwnerId);

446         /*
447          * Cross reference the namespace here, in order to
448          * generate External() statements
449          */
450         AcpiDmCrossReferenceNamespace (AcpiGbl_ParseOpRoot,
451             AcpiGbl_RootNode, OwnerId);

453         if (AslCompilerdebug)
454         {
455             AcpiDmDumpTree (AcpiGbl_ParseOpRoot);
456         }
```



```
458         /* Find possible calls to external control methods */

460         AcpiDmFindOrphanMethods (AcpiGbl_ParseOpRoot);

462         /*
463          * If we found any external control methods, we must reparse
464          * the entire tree with the new information (namely, the
465          * number of arguments per method)
466          */
467         if (AcpiDmGetExternalMethodCount ())
468         {
469             fprintf (stderr,
470                 "\nFound %u external control methods, "
471                 "reparsing with new information\n",
472                 AcpiDmGetExternalMethodCount ());

474             /* Reparse, rebuild namespace. no need to xref namespace */

476             AcpiPsDeleteParseTree (AcpiGbl_ParseOpRoot);
477             AcpiNsDeleteNamespaceSubtree (AcpiGbl_RootNode);

479             AcpiGbl_RootNode                    = NULL;
480             AcpiGbl_RootNodeStruct.Name.Integer = ACPI_ROOT_NAME;
481             AcpiGbl_RootNodeStruct.DescriptorType = ACPI_DESC_TYPE_NAMED;
482             AcpiGbl_RootNodeStruct.Type         = ACPI_TYPE_DEVICE;
483             AcpiGbl_RootNodeStruct.Parent       = NULL;
484             AcpiGbl_RootNodeStruct.Child        = NULL;
485             AcpiGbl_RootNodeStruct.Peer         = NULL;
486             AcpiGbl_RootNodeStruct.Object       = NULL;
487             AcpiGbl_RootNodeStruct.Flags        = 0;

489             Status = AcpiNsRootInitialize ();
490             AcpiDmAddExternalsToNamespace ();

492             /* Parse the table again. No need to reload it, however */

494             Status = AdParseTable (Table, NULL, FALSE, FALSE);
495             if (ACPI_FAILURE (Status))
496             {
497                 AcpiOsPrintf ("Could not parse ACPI tables, %s\n",
498                     AcpiFormatException (Status));
499                 goto Cleanup;
500             }

502             if (AslCompilerdebug)
503             {
504                 AcpiOsPrintf ("/**** After second load and resource conversion\n
505                 NsSetupNamespaceListing (File);
506                 NsDisplayNamespace ();
507                 AcpiOsPrintf ("*****/\n");

509                 AcpiDmDumpTree (AcpiGbl_ParseOpRoot);
510             }
511         }

513         /*
514          * Now that the namespace is finalized, we can perform namespace
515          * transforms.
516          *
517          * 1) Convert fixed-offset references to resource descriptors
518          *    to symbolic references (Note: modifies namespace)
519          */
520         AcpiDmConvertResourceIndexes (AcpiGbl_ParseOpRoot, AcpiGbl_RootNode);

522         /* Optional displays */
```

```
524         if (AcpiGbl_DbOpt_disasm)
525         {
526             /* This is the real disassembly */

528             AdDisplayTables (Filename, Table);

530             /* Dump hex table if requested (-vt) */

532             AcpiDmDumpDataTable (Table);

534             fprintf (stderr, "Disassembly completed\n");
535             fprintf (stderr, "ASL Output:    %s - %u bytes\n",
536                 DisasmFilename, AdGetFileSize (File));
537         }
538     }

540 Cleanup:

542     if (Table && !AcpiUtIsAmlTable (Table))
543     {
544         ACPI_FREE (Table);
545     }

547     if (OutToFile && File)
548     {
549         if (AslCompilerdebug) /* Display final namespace, with transforms */
550         {
551             NsSetupNamespaceListing (File);
552             NsDisplayNamespace ();
553         }

555         fclose (File);
556         AcpiOsRedirectOutput (stdout);
557     }

559     AcpiPsDeleteParseTree (AcpiGbl_ParseOpRoot);
560     AcpiGbl_ParseOpRoot = NULL;
561     return (Status);
562 }


565 /*****************************************************************************
566  *
567  * FUNCTION:    AdDisassemblerHeader
568  *
569  * PARAMETERS:  Filename            - Input file for the table
570  *
571  * RETURN:      None
572  *
573  * DESCRIPTION: Create the disassembler header, including ACPI CA signon with
574  *              current time and date.
575  *
576  *****************************************************************************/

578 void
579 AdDisassemblerHeader (
580     char                    *Filename)
581 {
582     time_t                  Timer;

584     time (&Timer);

586     /* Header and input table info */

588     AcpiOsPrintf ("/*\n");
```



```
589     AcpiOsPrintf (ACPI_COMMON_HEADER ("AML Disassembler", " * "));

591     AcpiOsPrintf (" * Disassembly of %s, %s", Filename, ctime (&Timer));
592     AcpiOsPrintf (" *\n");
593 }


596 /*****************************************************************************
597  *
598  * FUNCTION:    AdCreateTableHeader
599  *
600  * PARAMETERS:  Filename            - Input file for the table
601  *              Table               - Pointer to the raw table
602  *
603  * RETURN:      None
604  *
605  * DESCRIPTION: Create the ASL table header, including ACPI CA signon with
606  *              current time and date.
607  *
608  *****************************************************************************/

610 static void
611 AdCreateTableHeader (
612     char                    *Filename,
613     ACPI_TABLE_HEADER       *Table)
614 {
615     char                    *NewFilename;
616     UINT8                   Checksum;


619     /*
620      * Print file header and dump original table header
621      */
622     AdDisassemblerHeader (Filename);

624     AcpiOsPrintf (" * Original Table Header:\n");
625     AcpiOsPrintf (" *     Signature        \"%4.4s\"\n",    Table->Signature);
626     AcpiOsPrintf (" *     Length           0x%8.8X (%u)\n", Table->Length, Table

628     /* Print and validate the revision */

630     AcpiOsPrintf (" *     Revision         0x%2.2X",      Table->Revision);

632     switch (Table->Revision)
633     {
634     case 0:

636         AcpiOsPrintf (" **** Invalid Revision");
637         break;

639     case 1:

641         /* Revision of DSDT controls the ACPI integer width */

643         if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_DSDT))
644         {
645             AcpiOsPrintf (" **** 32-bit table (V1), no 64-bit math support");
646         }
647         break;

649     default:

651         break;
652     }
653     AcpiOsPrintf ("\n");
```

```
655     /* Print and validate the table checksum */

657     AcpiOsPrintf (" *     Checksum         0x%2.2X",        Table->Checksum);

659     Checksum = AcpiTbChecksum (ACPI_CAST_PTR (UINT8, Table), Table->Length);
660     if (Checksum)
661     {
662         AcpiOsPrintf (" **** Incorrect checksum, should be 0x%2.2X",
663             (UINT8) (Table->Checksum - Checksum));
664     }
665     AcpiOsPrintf ("\n");

667     AcpiOsPrintf (" *     OEM ID           \"%.6s\"\n",     Table->OemId);
668     AcpiOsPrintf (" *     OEM Table ID     \"%.8s\"\n",     Table->OemTableId);
669     AcpiOsPrintf (" *     OEM Revision     0x%8.8X (%u)\n", Table->OemRevision,
670     AcpiOsPrintf (" *     Compiler ID      \"%.4s\"\n",     Table->AslCompilerId
671     AcpiOsPrintf (" *     Compiler Version 0x%8.8X (%u)\n", Table->AslCompilerRe
672     AcpiOsPrintf (" */\n");

674     /* Create AML output filename based on input filename */

676     if (Filename)
677     {
678         NewFilename = FlGenerateFilename (Filename, "aml");
679     }
680     else
681     {
682         NewFilename = ACPI_ALLOCATE_ZEROED (9);
683         strncat (NewFilename, Table->Signature, 4);
684         strcat (NewFilename, ".aml");
685     }

687     /* Open the ASL definition block */

689     AcpiOsPrintf (
690         "DefinitionBlock (\"%s\", \"%4.4s\", %hu, \"%.6s\", \"%.8s\", 0x%8.8X)\n
691         NewFilename, Table->Signature, Table->Revision,
692         Table->OemId, Table->OemTableId, Table->OemRevision);

694     ACPI_FREE (NewFilename);
695 }


698 /******************************************************************************
699  *
700  * FUNCTION:    AdDisplayTables
701  *
702  * PARAMETERS:  Filename            - Input file for the table
703  *              Table               - Pointer to the raw table
704  *
705  * RETURN:      Status
706  *
707  * DESCRIPTION: Display (disassemble) loaded tables and dump raw tables
708  *
709  *****************************************************************************/

711 ACPI_STATUS
712 AdDisplayTables (
713     char                    *Filename,
714     ACPI_TABLE_HEADER       *Table)
715 {


718     if (!AcpiGbl_ParseOpRoot)
719     {
720         return (AE_NOT_EXIST);
```



```
721     }

723     if (!AcpiGbl_DbOpt_verbose)
724     {
725         AdCreateTableHeader (Filename, Table);
726     }

728     AcpiDmDisassemble (NULL, AcpiGbl_ParseOpRoot, ACPI_UINT32_MAX);

730     if (AcpiGbl_DbOpt_verbose)
731     {
732         AcpiOsPrintf ("\n\nTable Header:\n");
733         AcpiUtDebugDumpBuffer ((UINT8 *) Table, sizeof (ACPI_TABLE_HEADER),
734             DB_BYTE_DISPLAY, ACPI_UINT32_MAX);

736         AcpiOsPrintf ("Table Body (Length 0x%X)\n", Table->Length);
737         AcpiUtDebugDumpBuffer (((UINT8 *) Table + sizeof (ACPI_TABLE_HEADER)),
738             Table->Length, DB_BYTE_DISPLAY, ACPI_UINT32_MAX);
739     }

741     return (AE_OK);
742 }


745 /******************************************************************************
746  *
747  * FUNCTION:    AdGetLocalTables
748  *
749  * PARAMETERS:  Filename            - Not used
750  *              GetAllTables        - TRUE if all tables are desired
751  *
752  * RETURN:      Status
753  *
754  * DESCRIPTION: Get the ACPI tables from either memory or a file
755  *
756  *****************************************************************************/

758 ACPI_STATUS
759 AdGetLocalTables (
760     char                    *Filename,
761     BOOLEAN                 GetAllTables)
762 {
763     ACPI_STATUS             Status;
764     ACPI_TABLE_HEADER       TableHeader;
765     ACPI_TABLE_HEADER       *NewTable;
766     UINT32                  NumTables;
767     UINT32                  PointerSize;
768     UINT32                  TableIndex;


771     if (GetAllTables)
772     {
773         ACPI_MOVE_32_TO_32 (TableHeader.Signature, ACPI_SIG_RSDT);
774         AcpiOsTableOverride (&TableHeader, &NewTable);
775         if (!NewTable)
776         {
777             fprintf (stderr, "Could not obtain RSDT\n");
778             return (AE_NO_ACPI_TABLES);
779         }
780         else
781         {
782             AdWriteTable (NewTable, NewTable->Length,
783                 ACPI_SIG_RSDT, NewTable->OemTableId);
784         }

786         if (ACPI_COMPARE_NAME (NewTable->Signature, ACPI_SIG_RSDT))
```

```
787         {
788             PointerSize = sizeof (UINT32);
789         }
790         else
791         {
792             PointerSize = sizeof (UINT64);
793         }

795         /*
796          * Determine the number of tables pointed to by the RSDT/XSDT.
797          * This is defined by the ACPI Specification to be the number of
798          * pointers contained within the RSDT/XSDT. The size of the pointers
799          * is architecture-dependent.
800          */
801         NumTables = (NewTable->Length - sizeof (ACPI_TABLE_HEADER)) / PointerSiz
802         AcpiOsPrintf ("There are %u tables defined in the %4.4s\n\n",
803             NumTables, NewTable->Signature);

805         /* Get the FADT */

807         ACPI_MOVE_32_TO_32 (TableHeader.Signature, ACPI_SIG_FADT);
808         AcpiOsTableOverride (&TableHeader, &NewTable);
809         if (NewTable)
810         {
811             AdWriteTable (NewTable, NewTable->Length,
812                 ACPI_SIG_FADT, NewTable->OemTableId);
813         }
814         AcpiOsPrintf ("\n");

816         /* Don't bother with FACS, it is usually all zeros */
817     }

819     /* Always get the DSDT */

821     ACPI_MOVE_32_TO_32 (TableHeader.Signature, ACPI_SIG_DSDT);
822     AcpiOsTableOverride (&TableHeader, &NewTable);
823     if (NewTable)
824     {
825         AdWriteTable (NewTable, NewTable->Length,
826             ACPI_SIG_DSDT, NewTable->OemTableId);

828         /* Store DSDT in the Table Manager */

830         Status = AcpiTbStoreTable (0, NewTable, NewTable->Length,
831                     0, &TableIndex);
832         if (ACPI_FAILURE (Status))
833         {
834             fprintf (stderr, "Could not store DSDT\n");
835             return (AE_NO_ACPI_TABLES);
836         }
837     }
838     else
839     {
840         fprintf (stderr, "Could not obtain DSDT\n");
841         return (AE_NO_ACPI_TABLES);
842     }

844 #if 0
845     /* TBD: Future implementation */

847     AcpiOsPrintf ("\n");

849     /* Get all SSDTs */

851     ACPI_MOVE_32_TO_32 (TableHeader.Signature, ACPI_SIG_SSDT);
852     do
```



```
853     {
854         NewTable = NULL;
855         Status = AcpiOsTableOverride (&TableHeader, &NewTable);

857     } while (NewTable);
858 #endif

860     return (AE_OK);
861 }


864 /*****************************************************************************
865  *
866  * FUNCTION:    AdParseTable
867  *
868  * PARAMETERS:  Table               - Pointer to the raw table
869  *              OwnerId             - Returned OwnerId of the table
870  *              LoadTable           - If add table to the global table list
871  *              External            - If this is an external table
872  *
873  * RETURN:      Status
874  *
875  * DESCRIPTION: Parse the DSDT.
876  *
877  ****************************************************************************/

879 ACPI_STATUS
880 AdParseTable (
881     ACPI_TABLE_HEADER       *Table,
882     ACPI_OWNER_ID           *OwnerId,
883     BOOLEAN                 LoadTable,
884     BOOLEAN                 External)
885 {
886     ACPI_STATUS             Status = AE_OK;
887     ACPI_WALK_STATE         *WalkState;
888     UINT8                   *AmlStart;
889     UINT32                  AmlLength;
890     UINT32                  TableIndex;


893     if (!Table)
894     {
895         return (AE_NOT_EXIST);
896     }

898     /* Pass 1:  Parse everything except control method bodies */

900     fprintf (stderr, "Pass 1 parse of [%4.4s]\n", (char *) Table->Signature);

902     AmlLength = Table->Length - sizeof (ACPI_TABLE_HEADER);
903     AmlStart = ((UINT8 *) Table + sizeof (ACPI_TABLE_HEADER));

905     /* Create the root object */

907     AcpiGbl_ParseOpRoot = AcpiPsCreateScopeOp ();
908     if (!AcpiGbl_ParseOpRoot)
909     {
910         return (AE_NO_MEMORY);
911     }

913     /* Create and initialize a new walk state */

915     WalkState = AcpiDsCreateWalkState (0,
916                     AcpiGbl_ParseOpRoot, NULL, NULL);
917     if (!WalkState)
918     {
```

```
919         return (AE_NO_MEMORY);
920     }

922     Status = AcpiDsInitAmlWalk (WalkState, AcpiGbl_ParseOpRoot,
923                 NULL, AmlStart, AmlLength, NULL, ACPI_IMODE_LOAD_PASS1);
924     if (ACPI_FAILURE (Status))
925     {
926         return (Status);
927     }

929     WalkState->ParseFlags &= ~ACPI_PARSE_DELETE_TREE;
930     WalkState->ParseFlags |= ACPI_PARSE_DISASSEMBLE;

932     Status = AcpiPsParseAml (WalkState);
933     if (ACPI_FAILURE (Status))
934     {
935         return (Status);
936     }

938     /* If LoadTable is FALSE, we are parsing the last loaded table */

940     TableIndex = AcpiGbl_RootTableList.CurrentTableCount - 1;

942     /* Pass 2 */

944     if (LoadTable)
945     {
946         Status = AcpiTbStoreTable ((ACPI_PHYSICAL_ADDRESS) Table, Table,
947                     Table->Length, ACPI_TABLE_ORIGIN_ALLOCATED, &TableIndex);
948         if (ACPI_FAILURE (Status))
949         {
950             return (Status);
951         }
952         Status = AcpiTbAllocateOwnerId (TableIndex);
953         if (ACPI_FAILURE (Status))
954         {
955             return (Status);
956         }
957         if (OwnerId)
958         {
959             Status = AcpiTbGetOwnerId (TableIndex, OwnerId);
960             if (ACPI_FAILURE (Status))
961             {
962                 return (Status);
963             }
964         }
965     }

967     fprintf (stderr, "Pass 2 parse of [%4.4s]\n", (char *) Table->Signature);

969     Status = AcpiNsOneCompleteParse (ACPI_IMODE_LOAD_PASS2, TableIndex, NULL);
970     if (ACPI_FAILURE (Status))
971     {
972         return (Status);
973     }

975     /* No need to parse control methods of external table */

977     if (External)
978     {
979         return (AE_OK);
980     }

982     /* Pass 3: Parse control methods and link their parse trees into the main pa

984     fprintf (stderr, "Parsing Deferred Opcodes (Methods/Buffers/Packages/Regions
```



```
985     Status = AcpiDmParseDeferredOps (AcpiGbl_ParseOpRoot);
986     fprintf (stderr, "\n");

988     /* Process Resource Templates */

990     AcpiDmFindResources (AcpiGbl_ParseOpRoot);

992     fprintf (stderr, "Parsing completed\n");
993     return (AE_OK);
994 }
```

```
**********************************************************
   29204 Thu Dec 26 13:48:23 2013
new/usr/src/common/acpica/common/adwalk.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: adwalk - Application-level disassembler parse tree walk routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdisasm.h"
#include "acdispat.h"
#include "acnamesp.h"
#include "acapps.h"


#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("adwalk")

/*
```



```
 * aslmap - opcode mappings and reserved method names
 */
ACPI_OBJECT_TYPE
AslMapNamedOpcodeToDataType (
    UINT16                  Opcode);

/* Local prototypes */

static ACPI_STATUS
AcpiDmFindOrphanDescending (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
AcpiDmDumpDescending (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
AcpiDmXrefDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
AcpiDmCommonAscendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
AcpiDmLoadDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static UINT32
AcpiDmInspectPossibleArgs (
    UINT32                  CurrentOpArgCount,
    UINT32                  TargetCount,
    ACPI_PARSE_OBJECT       *Op);

static ACPI_STATUS
AcpiDmResourceDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpTree
 *
 * PARAMETERS:  Origin              - Starting object
 *
 * RETURN:      None
 *
 * DESCRIPTION: Parse tree walk to format and output the nodes
 *
 ******************************************************************************/

void
AcpiDmDumpTree (
    ACPI_PARSE_OBJECT       *Origin)
```

```
125 {
126     ACPI_OP_WALK_INFO       Info;


129     if (!Origin)
130     {
131         return;
132     }

134     AcpiOsPrintf ("/*\nAML Parse Tree\n\n");
135     Info.Flags = 0;
136     Info.Count = 0;
137     Info.Level = 0;
138     Info.WalkState = NULL;
139     AcpiDmWalkParseTree (Origin, AcpiDmDumpDescending, NULL, &Info);
140     AcpiOsPrintf ("*/\n\n");
141 }


144 /*******************************************************************************
145  *
146  * FUNCTION:    AcpiDmFindOrphanMethods
147  *
148  * PARAMETERS:  Origin              - Starting object
149  *
150  * RETURN:      None
151  *
152  * DESCRIPTION: Parse tree walk to find "orphaned" method invocations -- methods
153  *              that are not resolved in the namespace
154  *
155  ******************************************************************************/

157 void
158 AcpiDmFindOrphanMethods (
159     ACPI_PARSE_OBJECT       *Origin)
160 {
161     ACPI_OP_WALK_INFO       Info;


164     if (!Origin)
165     {
166         return;
167     }

169     Info.Flags = 0;
170     Info.Level = 0;
171     Info.WalkState = NULL;
172     AcpiDmWalkParseTree (Origin, AcpiDmFindOrphanDescending, NULL, &Info);
173 }


176 /*******************************************************************************
177  *
178  * FUNCTION:    AcpiDmFinishNamespaceLoad
179  *
180  * PARAMETERS:  ParseTreeRoot       - Root of the parse tree
181  *              NamespaceRoot       - Root of the internal namespace
182  *              OwnerId             - OwnerId of the table to be disassembled
183  *
184  * RETURN:      None
185  *
186  * DESCRIPTION: Load all namespace items that are created within control
187  *              methods. Used before namespace cross reference
188  *
189  ******************************************************************************/
```



```
191 void
192 AcpiDmFinishNamespaceLoad (
193     ACPI_PARSE_OBJECT       *ParseTreeRoot,
194     ACPI_NAMESPACE_NODE     *NamespaceRoot,
195     ACPI_OWNER_ID           OwnerId)
196 {
197     ACPI_STATUS             Status;
198     ACPI_OP_WALK_INFO       Info;
199     ACPI_WALK_STATE         *WalkState;


202     if (!ParseTreeRoot)
203     {
204         return;
205     }

207     /* Create and initialize a new walk state */

209     WalkState = AcpiDsCreateWalkState (OwnerId, ParseTreeRoot, NULL, NULL);
210     if (!WalkState)
211     {
212         return;
213     }

215     Status = AcpiDsScopeStackPush (NamespaceRoot, NamespaceRoot->Type, WalkState
216     if (ACPI_FAILURE (Status))
217     {
218         return;
219     }

221     Info.Flags = 0;
222     Info.Level = 0;
223     Info.WalkState = WalkState;
224     AcpiDmWalkParseTree (ParseTreeRoot, AcpiDmLoadDescendingOp,
225         AcpiDmCommonAscendingOp, &Info);
226     ACPI_FREE (WalkState);
227 }


230 /*******************************************************************************
231  *
232  * FUNCTION:    AcpiDmCrossReferenceNamespace
233  *
234  * PARAMETERS:  ParseTreeRoot       - Root of the parse tree
235  *              NamespaceRoot       - Root of the internal namespace
236  *              OwnerId             - OwnerId of the table to be disassembled
237  *
238  * RETURN:      None
239  *
240  * DESCRIPTION: Cross reference the namespace to create externals
241  *
242  ******************************************************************************/

244 void
245 AcpiDmCrossReferenceNamespace (
246     ACPI_PARSE_OBJECT       *ParseTreeRoot,
247     ACPI_NAMESPACE_NODE     *NamespaceRoot,
248     ACPI_OWNER_ID           OwnerId)
249 {
250     ACPI_STATUS             Status;
251     ACPI_OP_WALK_INFO       Info;
252     ACPI_WALK_STATE         *WalkState;


255     if (!ParseTreeRoot)
256     {
```

```
257         return;
258     }

260     /* Create and initialize a new walk state */

262     WalkState = AcpiDsCreateWalkState (OwnerId, ParseTreeRoot, NULL, NULL);
263     if (!WalkState)
264     {
265         return;
266     }

268     Status = AcpiDsScopeStackPush (NamespaceRoot, NamespaceRoot->Type, WalkState
269     if (ACPI_FAILURE (Status))
270     {
271         return;
272     }

274     Info.Flags = 0;
275     Info.Level = 0;
276     Info.WalkState = WalkState;
277     AcpiDmWalkParseTree (ParseTreeRoot, AcpiDmXrefDescendingOp,
278         AcpiDmCommonAscendingOp, &Info);
279     ACPI_FREE (WalkState);
280 }


283 /*******************************************************************************
284  *
285  * FUNCTION:    AcpiDmConvertResourceIndexes
286  *
287  * PARAMETERS:  ParseTreeRoot       - Root of the parse tree
288  *              NamespaceRoot       - Root of the internal namespace
289  *
290  * RETURN:      None
291  *
292  * DESCRIPTION: Convert fixed-offset references to resource descriptors to
293  *              symbolic references. Should only be called after namespace has
294  *              been cross referenced.
295  *
296  ******************************************************************************/

298 void
299 AcpiDmConvertResourceIndexes (
300     ACPI_PARSE_OBJECT       *ParseTreeRoot,
301     ACPI_NAMESPACE_NODE     *NamespaceRoot)
302 {
303     ACPI_STATUS             Status;
304     ACPI_OP_WALK_INFO       Info;
305     ACPI_WALK_STATE         *WalkState;


308     if (!ParseTreeRoot)
309     {
310         return;
311     }

313     /* Create and initialize a new walk state */

315     WalkState = AcpiDsCreateWalkState (0, ParseTreeRoot, NULL, NULL);
316     if (!WalkState)
317     {
318         return;
319     }

321     Status = AcpiDsScopeStackPush (NamespaceRoot, NamespaceRoot->Type, WalkState
322     if (ACPI_FAILURE (Status))
```



```
323     {
324         return;
325     }

327     Info.Flags = 0;
328     Info.Level = 0;
329     Info.WalkState = WalkState;
330     AcpiDmWalkParseTree (ParseTreeRoot, AcpiDmResourceDescendingOp,
331         AcpiDmCommonAscendingOp, &Info);
332     ACPI_FREE (WalkState);
333     return;
334 }


337 /*******************************************************************************
338  *
339  * FUNCTION:    AcpiDmDumpDescending
340  *
341  * PARAMETERS:  ASL_WALK_CALLBACK
342  *
343  * RETURN:      Status
344  *
345  * DESCRIPTION: Format and print contents of one parse Op.
346  *
347  ******************************************************************************/

349 static ACPI_STATUS
350 AcpiDmDumpDescending (
351     ACPI_PARSE_OBJECT       *Op,
352     UINT32                  Level,
353     void                    *Context)
354 {
355     ACPI_OP_WALK_INFO       *Info = Context;
356     char                    *Path;


359     if (!Op)
360     {
361         return (AE_OK);
362     }

364     /* Most of the information (count, level, name) here */

366     Info->Count++;
367     AcpiOsPrintf ("% 5d [%2.2d] ", Info->Count, Level);
368     AcpiDmIndent (Level);
369     AcpiOsPrintf ("%-28s", AcpiPsGetOpcodeName (Op->Common.AmlOpcode));

371     /* Extra info is helpful */

373     switch (Op->Common.AmlOpcode)
374     {
375     case AML_BYTE_OP:

377         AcpiOsPrintf ("%2.2X", (UINT32) Op->Common.Value.Integer);
378         break;

380     case AML_WORD_OP:

382         AcpiOsPrintf ("%4.4X", (UINT32) Op->Common.Value.Integer);
383         break;

385     case AML_DWORD_OP:

387         AcpiOsPrintf ("%8.8X", (UINT32) Op->Common.Value.Integer);
388         break;
```

```
390     case AML_QWORD_OP:

392         AcpiOsPrintf ("%8.8X%8.8X", ACPI_FORMAT_UINT64 (Op->Common.Value.Integer
393         break;

395     case AML_INT_NAMEPATH_OP:

397         if (Op->Common.Value.String)
398         {
399             AcpiNsExternalizeName (ACPI_UINT32_MAX, Op->Common.Value.String,
400                         NULL, &Path);
401             AcpiOsPrintf ("%s %p", Path, Op->Common.Node);
402             ACPI_FREE (Path);
403         }
404         else
405         {
406             AcpiOsPrintf ("[NULL]");
407         }
408         break;

410     case AML_NAME_OP:
411     case AML_METHOD_OP:
412     case AML_DEVICE_OP:
413     case AML_INT_NAMEDFIELD_OP:

415         AcpiOsPrintf ("%4.4s", ACPI_CAST_PTR (char, &Op->Named.Name));
416         break;

418     default:

420         break;
421     }

423     AcpiOsPrintf ("\n");
424     return (AE_OK);
425 }


428 /*******************************************************************************
429  *
430  * FUNCTION:    AcpiDmFindOrphanDescending
431  *
432  * PARAMETERS:  ASL_WALK_CALLBACK
433  *
434  * RETURN:      Status
435  *
436  * DESCRIPTION: Check namepath Ops for orphaned method invocations
437  *
438  * Note: Experimental.
439  *
440  ******************************************************************************/

442 static ACPI_STATUS
443 AcpiDmFindOrphanDescending (
444     ACPI_PARSE_OBJECT      *Op,
445     UINT32                 Level,
446     void                   *Context)
447 {
448     const ACPI_OPCODE_INFO  *OpInfo;
449     ACPI_PARSE_OBJECT      *ChildOp;
450     ACPI_PARSE_OBJECT      *NextOp;
451     ACPI_PARSE_OBJECT      *ParentOp;
452     UINT32                 ArgCount;
```



```
455     if (!Op)
456     {
457         return (AE_OK);
458     }

460     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

462     switch (Op->Common.AmlOpcode)
463     {
464 #ifdef ACPI_UNDER_DEVELOPMENT
465     case AML_ADD_OP:

467         ChildOp = Op->Common.Value.Arg;
468         if ((ChildOp->Common.AmlOpcode == AML_INT_NAMEPATH_OP) &&
469             !ChildOp->Common.Node)
470         {
471             AcpiNsExternalizeName (ACPI_UINT32_MAX, ChildOp->Common.Value.String
472                 NULL, &Path);
473             AcpiOsPrintf ("/* %-16s A-NAMEPATH: %s  */\n",
474                 Op->Common.AmlOpName, Path);
475             ACPI_FREE (Path);

477             NextOp = Op->Common.Next;
478             if (!NextOp)
479             {
480                 /* This NamePath has no args, assume it is an integer */

482                 AcpiDmAddOpToExternalList (ChildOp,
483                     ChildOp->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
484                 return (AE_OK);
485             }

487             ArgCount = AcpiDmInspectPossibleArgs (3, 1, NextOp);
488             AcpiOsPrintf ("/* A-CHILDREN: %u Actual %u */\n",
489                 ArgCount, AcpiDmCountChildren (Op));

491             if (ArgCount < 1)
492             {
493                 /* One Arg means this is just a Store(Name,Target) */

495                 AcpiDmAddOpToExternalList (ChildOp,
496                     ChildOp->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
497                 return (AE_OK);
498             }

500             AcpiDmAddOpToExternalList (ChildOp,
501                 ChildOp->Common.Value.String, ACPI_TYPE_METHOD, ArgCount, 0);
502         }
503         break;
504 #endif

506     case AML_STORE_OP:

508         ChildOp = Op->Common.Value.Arg;
509         if ((ChildOp->Common.AmlOpcode == AML_INT_NAMEPATH_OP) &&
510             !ChildOp->Common.Node)
511         {
512             NextOp = Op->Common.Next;
513             if (!NextOp)
514             {
515                 /* This NamePath has no args, assume it is an integer */

517                 AcpiDmAddOpToExternalList (ChildOp,
518                     ChildOp->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
519                 return (AE_OK);
520             }
```

```
            ArgCount = AcpiDmInspectPossibleArgs (2, 1, NextOp);
            if (ArgCount <= 1)
            {
                /* One Arg means this is just a Store(Name,Target) */

                AcpiDmAddOpToExternalList (ChildOp,
                    ChildOp->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
                return (AE_OK);
            }

            AcpiDmAddOpToExternalList (ChildOp,
                ChildOp->Common.Value.String, ACPI_TYPE_METHOD, ArgCount, 0);
        }
        break;

    case AML_INT_NAMEPATH_OP:

        /* Must examine parent to see if this namepath is an argument */

        ParentOp = Op->Common.Parent;
        OpInfo = AcpiPsGetOpcodeInfo (ParentOp->Common.AmlOpcode);

        if ((OpInfo->Class != AML_CLASS_EXECUTE) &&
            (OpInfo->Class != AML_CLASS_CREATE) &&
            (OpInfo->ObjectType != ACPI_TYPE_LOCAL_ALIAS) &&
            (ParentOp->Common.AmlOpcode != AML_INT_METHODCALL_OP) &&
            !Op->Common.Node)
        {
            ArgCount = AcpiDmInspectPossibleArgs (0, 0, Op->Common.Next);

            /*
             * Check if namepath is a predicate for if/while or lone parameter t
             * a return.
             */
            if (ArgCount == 0)
            {
                if (((ParentOp->Common.AmlOpcode == AML_IF_OP) ||
                     (ParentOp->Common.AmlOpcode == AML_WHILE_OP) ||
                     (ParentOp->Common.AmlOpcode == AML_RETURN_OP)) &&

                     /* And namepath is the first argument */
                     (ParentOp->Common.Value.Arg == Op))
                {
                    AcpiDmAddOpToExternalList (Op,
                        Op->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
                    break;
                }
            }

            /*
             * This is a standalone namestring (not a parameter to another
             * operator) - it *must* be a method invocation, nothing else is
             * grammatically possible.
             */
            AcpiDmAddOpToExternalList (Op,
                Op->Common.Value.String, ACPI_TYPE_METHOD, ArgCount, 0);
        }
        break;

    default:

        break;
    }

    return (AE_OK);
```



```
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmLoadDescendingOp
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Descending handler for namespace control method object load
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiDmLoadDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_OP_WALK_INFO       *Info = Context;
    const ACPI_OPCODE_INFO  *OpInfo;
    ACPI_WALK_STATE         *WalkState;
    ACPI_OBJECT_TYPE        ObjectType;
    ACPI_STATUS             Status;
    char                    *Path = NULL;
    ACPI_PARSE_OBJECT       *NextOp;
    ACPI_NAMESPACE_NODE     *Node;
    char                    FieldPath[5];
    BOOLEAN                 PreDefined = FALSE;
    UINT8                   PreDefineIndex = 0;


    WalkState = Info->WalkState;
    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
    ObjectType = OpInfo->ObjectType;
    ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);

    /* Only interested in operators that create new names */

    if (!(OpInfo->Flags & AML_NAMED) &&
        !(OpInfo->Flags & AML_CREATE))
    {
        goto Exit;
    }

    /* Get the NamePath from the appropriate place */

    if (OpInfo->Flags & AML_NAMED)
    {
        /* For all named operators, get the new name */

        Path = (char *) Op->Named.Path;

        if (!Path && Op->Common.AmlOpcode == AML_INT_NAMEDFIELD_OP)
        {
            *ACPI_CAST_PTR (UINT32, &FieldPath[0]) = Op->Named.Name;
            FieldPath[4] = 0;
            Path = FieldPath;
        }
    }
    else if (OpInfo->Flags & AML_CREATE)
    {
        /* New name is the last child */
```

```
653         NextOp = Op->Common.Value.Arg;

655         while (NextOp->Common.Next)
656         {
657             NextOp = NextOp->Common.Next;
658         }
659         Path = NextOp->Common.Value.String;
660     }

662     if (!Path)
663     {
664         goto Exit;
665     }

667     /* Insert the name into the namespace */

669     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
670                 ACPI_IMODE_LOAD_PASS2, ACPI_NS_DONT_OPEN_SCOPE,
671                 WalkState, &Node);

673     Op->Common.Node = Node;

675     if (ACPI_SUCCESS (Status))
676     {
677         /* Check if it's a predefined node */

679         while (AcpiGbl_PreDefinedNames[PreDefineIndex].Name)
680         {
681             if (ACPI_COMPARE_NAME (Node->Name.Ascii,
682                 AcpiGbl_PreDefinedNames[PreDefineIndex].Name))
683             {
684                 PreDefined = TRUE;
685                 break;
686             }

688             PreDefineIndex++;
689         }

691         /*
692          * Set node owner id if it satisfies all the following conditions:
693          * 1) Not a predefined node, _SB_ etc
694          * 2) Not the root node
695          * 3) Not a node created by Scope
696          */

698         if (!PreDefined && Node != AcpiGbl_RootNode &&
699             Op->Common.AmlOpcode != AML_SCOPE_OP)
700         {
701             Node->OwnerId = WalkState->OwnerId;
702         }
703     }


706 Exit:

708     if (AcpiNsOpensScope (ObjectType))
709     {
710         if (Op->Common.Node)
711         {
712             Status = AcpiDsScopeStackPush (Op->Common.Node, ObjectType, WalkStat
713             if (ACPI_FAILURE (Status))
714             {
715                 return (Status);
716             }
717         }
718     }
```



```
720     return (AE_OK);
721 }


724 /*******************************************************************************
725  *
726  * FUNCTION:    AcpiDmXrefDescendingOp
727  *
728  * PARAMETERS:  ASL_WALK_CALLBACK
729  *
730  * RETURN:      Status
731  *
732  * DESCRIPTION: Descending handler for namespace cross reference
733  *
734  ******************************************************************************/

736 static ACPI_STATUS
737 AcpiDmXrefDescendingOp (
738     ACPI_PARSE_OBJECT       *Op,
739     UINT32                  Level,
740     void                    *Context)
741 {
742     ACPI_OP_WALK_INFO       *Info = Context;
743     const ACPI_OPCODE_INFO  *OpInfo;
744     ACPI_WALK_STATE         *WalkState;
745     ACPI_OBJECT_TYPE        ObjectType;
746     ACPI_OBJECT_TYPE        ObjectType2;
747     ACPI_STATUS             Status;
748     char                    *Path = NULL;
749     ACPI_PARSE_OBJECT       *NextOp;
750     ACPI_NAMESPACE_NODE     *Node;
751     ACPI_OPERAND_OBJECT     *Object;
752     UINT32                  ParamCount = 0;
753     char                    *Pathname;


756     WalkState = Info->WalkState;
757     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
758     ObjectType = OpInfo->ObjectType;
759     ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);

761     if ((!(OpInfo->Flags & AML_NAMED)) &&
762         (!(OpInfo->Flags & AML_CREATE)) &&
763         (Op->Common.AmlOpcode != AML_INT_NAMEPATH_OP))
764     {
765         goto Exit;
766     }

768     /* Get the NamePath from the appropriate place */

770     if (OpInfo->Flags & AML_NAMED)
771     {
772         /*
773          * Only these two operators (Alias, Scope) refer to an existing
774          * name, it is the first argument
775          */
776         if (Op->Common.AmlOpcode == AML_ALIAS_OP)
777         {
778             ObjectType = ACPI_TYPE_ANY;

780             NextOp = Op->Common.Value.Arg;
781             NextOp = NextOp->Common.Value.Arg;
782             if (NextOp->Common.AmlOpcode == AML_INT_NAMEPATH_OP)
783             {
784                 Path = NextOp->Common.Value.String;
```

```
785             }
786         }
787         else if (Op->Common.AmlOpcode == AML_SCOPE_OP)
788         {
789             Path = (char *) Op->Named.Path;
790         }
791     }
792     else if (OpInfo->Flags & AML_CREATE)
793     {
794         /* Referenced Buffer Name is the first child */

796         ObjectType = ACPI_TYPE_BUFFER; /* Change from TYPE_BUFFER_FIELD */

798         NextOp = Op->Common.Value.Arg;
799         if (NextOp->Common.AmlOpcode == AML_INT_NAMEPATH_OP)
800         {
801             Path = NextOp->Common.Value.String;
802         }
803     }
804     else
805     {
806         Path = Op->Common.Value.String;
807     }

809     if (!Path)
810     {
811         goto Exit;
812     }

814     /*
815      * Lookup the name in the namespace. Name must exist at this point, or it
816      * is an invalid reference.
817      *
818      * The namespace is also used as a lookup table for references to resource
819      * descriptors and the fields within them.
820      */
821     Node = NULL;
822     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ACPI_TYPE_ANY,
823                 ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPEN_SC
824                 WalkState, &Node);
825     if (ACPI_SUCCESS (Status) && (Node->Flags & ANOBJ_IS_EXTERNAL))
826     {
827         /* Node was created by an External() statement */

829         Status = AE_NOT_FOUND;
830     }

832     if (ACPI_FAILURE (Status))
833     {
834         if (Status == AE_NOT_FOUND)
835         {
836             /*
837              * Add this symbol as an external declaration, except if the
838              * parent is a CondRefOf operator. For this operator, we do not
839              * need an external, nor do we want one, since this can cause
840              * disassembly problems if the symbol is actually a control
841              * method.
842              */
843             if (!(Op->Asl.Parent &&
844                 (Op->Asl.Parent->Asl.AmlOpcode == AML_COND_REF_OF_OP)))
845             {
846                 if (Node)
847                 {
848                     AcpiDmAddNodeToExternalList (Node,
849                         (UINT8) ObjectType, 0, 0);
850                 }
```



```
851                 else
852                 {
853                     AcpiDmAddOpToExternalList (Op, Path,
854                         (UINT8) ObjectType, 0, 0);
855                 }
856             }
857         }
858     }

860     /*
861      * Found the node, but check if it came from an external table.
862      * Add it to external list. Note: Node->OwnerId == 0 indicates
863      * one of the built-in ACPI Names (_OS_ etc.) which can safely
864      * be ignored.
865      */
866     else if (Node->OwnerId &&
867             (WalkState->OwnerId != Node->OwnerId))
868     {
869         ObjectType2 = ObjectType;

871         Object = AcpiNsGetAttachedObject (Node);
872         if (Object)
873         {
874             ObjectType2 = Object->Common.Type;
875             if (ObjectType2 == ACPI_TYPE_METHOD)
876             {
877                 ParamCount = Object->Method.ParamCount;
878             }
879         }

881         Pathname = AcpiNsGetExternalPathname (Node);
882         if (!Pathname)
883         {
884             return (AE_NO_MEMORY);
885         }

887         AcpiDmAddNodeToExternalList (Node, (UINT8) ObjectType2,
888             ParamCount, ACPI_EXT_RESOLVED_REFERENCE);

890         ACPI_FREE (Pathname);
891         Op->Common.Node = Node;
892     }
893     else
894     {
895         Op->Common.Node = Node;
896     }


899 Exit:
900     /* Open new scope if necessary */

902     if (AcpiNsOpensScope (ObjectType))
903     {
904         if (Op->Common.Node)
905         {
906             Status = AcpiDsScopeStackPush (Op->Common.Node, ObjectType, WalkStat
907             if (ACPI_FAILURE (Status))
908             {
909                 return (Status);
910             }
911         }
912     }

914     return (AE_OK);
915 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiDmResourceDescendingOp
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      None
 *
 * DESCRIPTION: Process one parse op during symbolic resource index conversion.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiDmResourceDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_OP_WALK_INFO       *Info = Context;
    const ACPI_OPCODE_INFO  *OpInfo;
    ACPI_WALK_STATE         *WalkState;
    ACPI_OBJECT_TYPE        ObjectType;
    ACPI_STATUS             Status;


    WalkState = Info->WalkState;
    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    /* Open new scope if necessary */

    ObjectType = OpInfo->ObjectType;
    if (AcpiNsOpensScope (ObjectType))
    {
        if (Op->Common.Node)
        {

            Status = AcpiDsScopeStackPush (Op->Common.Node, ObjectType, WalkStat
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
        }
    }

    /*
     * Check if this operator contains a reference to a resource descriptor.
     * If so, convert the reference into a symbolic reference.
     */
    AcpiDmCheckResourceReference (Op, WalkState);
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmCommonAscendingOp
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      None
 *
 * DESCRIPTION: Ascending handler for combined parse/namespace walks. Closes
 *              scope if necessary.
 *
 ******************************************************************************/
```



```
static ACPI_STATUS
AcpiDmCommonAscendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_OP_WALK_INFO       *Info = Context;
    const ACPI_OPCODE_INFO  *OpInfo;
    ACPI_OBJECT_TYPE        ObjectType;


    /* Close scope if necessary */

    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
    ObjectType = OpInfo->ObjectType;
    ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);

    if (AcpiNsOpensScope (ObjectType))
    {
        (void) AcpiDsScopeStackPop (Info->WalkState);
    }

    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmInspectPossibleArgs
 *
 * PARAMETERS:  CurrentOpArgCount   - Which arg of the current op was the
 *                                    possible method invocation found
 *              TargetCount         - Number of targets (0,1,2) for this op
 *              Op                  - Parse op
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Examine following args and next ops for possible arguments
 *              for an unrecognized method invocation.
 *
 ******************************************************************************/

static UINT32
AcpiDmInspectPossibleArgs (
    UINT32                  CurrentOpArgCount,
    UINT32                  TargetCount,
    ACPI_PARSE_OBJECT       *Op)
{
    const ACPI_OPCODE_INFO  *OpInfo;
    UINT32                  i;
    UINT32                  Last = 0;
    UINT32                  Lookahead;


    Lookahead = (ACPI_METHOD_NUM_ARGS + TargetCount) - CurrentOpArgCount;

    /* Lookahead for the maximum number of possible arguments */

    for (i = 0; i < Lookahead; i++)
    {
        if (!Op)
        {
            break;
        }
```

```
1049         OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
1050
1051         /*
1052          * Any one of these operators is "very probably" not a method arg
1053          */
1054         if ((Op->Common.AmlOpcode == AML_STORE_OP) ||
1055             (Op->Common.AmlOpcode == AML_NOTIFY_OP))
1056         {
1057             break;
1058         }
1059
1060         if ((OpInfo->Class != AML_CLASS_EXECUTE) &&
1061             (OpInfo->Class != AML_CLASS_CONTROL))
1062         {
1063             Last = i+1;
1064         }
1065
1066         Op = Op->Common.Next;
1067     }
1068
1069     return (Last);
1070 }
```

```
**********************************************************
   28510 Thu Dec 26 13:48:23 2013
new/usr/src/common/acpica/common/ahpredef.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ahpredef - Table of all known ACPI predefined names
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

/*
 * iASL only needs a partial table (short descriptions only).
 * AcpiHelp needs the full table.
 */
#ifdef ACPI_ASL_COMPILER
#define AH_PREDEF(Name, ShortDesc, LongDesc) {Name, ShortDesc}
#else
#define AH_PREDEF(Name, ShortDesc, LongDesc) {Name, ShortDesc, LongDesc}
#endif

/*
 * Predefined ACPI names, with short description and return value.
 * This table was extracted directly from the ACPI specification.
 */
const AH_PREDEFINED_NAME    AslPredefinedInfo[] =
```



```
{
    AH_PREDEF ("_ACx",    "Active Cooling", "Returns the active cooling policy t
    AH_PREDEF ("_ADR",    "Address", "Returns address of a device on parent bus,
    AH_PREDEF ("_AEI",    "ACPI Event Interrupts", "Returns a list of GPIO event
    AH_PREDEF ("_ALC",    "Ambient Light Chromaticity", "Returns the ambient lig
    AH_PREDEF ("_ALI",    "Ambient Light Illuminance", "Returns the ambient ligh
    AH_PREDEF ("_ALN",    "Alignment", "Base alignment, Resource Descriptor fiel
    AH_PREDEF ("_ALP",    "Ambient Light Polling", "Returns the ambient light se
    AH_PREDEF ("_ALR",    "Ambient Light Response", "Returns the ambient light b
    AH_PREDEF ("_ALT",    "Ambient Light Temperature", "Returns the ambient ligh
    AH_PREDEF ("_ALx",    "Active List", "Returns a list of active cooling devic
    AH_PREDEF ("_ART",    "Active Cooling Relationship Table", "Returns thermal
    AH_PREDEF ("_ASI",    "Address Space Id", "Resource Descriptor field"),
    AH_PREDEF ("_ASZ",    "Access Size", "Resource Descriptor field"),
    AH_PREDEF ("_ATT",    "Type-Specific Attribute", "Resource Descriptor field"
    AH_PREDEF ("_BAS",    "Base Address", "Range base address, Resource Descript
    AH_PREDEF ("_BBN",    "BIOS Bus Number", "Returns the PCI bus number returne
    AH_PREDEF ("_BCL",    "Brightness Control Levels", "Returns a list of suppor
    AH_PREDEF ("_BCM",    "Brightness Control Method", "Sets the brightness leve
    AH_PREDEF ("_BCT",    "Battery Charge Time", "Returns time remaining to comp
    AH_PREDEF ("_BDN",    "BIOS Dock Name", "Returns the Dock ID returned by the
    AH_PREDEF ("_BFS",    "Back From Sleep", "Inform AML of a wake event"),
    AH_PREDEF ("_BIF",    "Battery Information", "Returns a Control Method Batte
    AH_PREDEF ("_BIX",    "Battery Information Extended", "Returns a Control Met
    AH_PREDEF ("_BLT",    "Battery Level Threshold", "Set battery level threshol
    AH_PREDEF ("_BM_",    "Bus Master", "Resource Descriptor field"),
    AH_PREDEF ("_BMA",    "Battery Measurement Averaging Interval", "Sets batter
    AH_PREDEF ("_BMC",    "Battery Maintenance Control", "Sets battery maintenan
    AH_PREDEF ("_BMD",    "Battery Maintenance Data", "Returns battery maintenan
    AH_PREDEF ("_BMS",    "Battery Measurement Sampling Time", "Sets the battery
    AH_PREDEF ("_BQC",    "Brightness Query Current", "Returns the current displ
    AH_PREDEF ("_BST",    "Battery Status", "Returns a Control Method Battery st
    AH_PREDEF ("_BTM",    "Battery Time", "Returns the battery runtime"),
    AH_PREDEF ("_BTP",    "Battery Trip Point", "Sets a Control Method Battery t
    AH_PREDEF ("_CBA",    "Configuration Base Address", "Sets the base address f
    AH_PREDEF ("_CDM",    "Clock Domain", "Returns a logical processor's clock d
    AH_PREDEF ("_CID",    "Compatible ID", "Returns a device's Plug and Play Com
    AH_PREDEF ("_CLS",    "Class Code", "Returns PCI class code and subclass"),
    AH_PREDEF ("_CPC",    "Continuous Performance Control", "Returns a list of p
    AH_PREDEF ("_CRS",    "Current Resource Settings", "Returns the current reso
    AH_PREDEF ("_CRT",    "Critical Temperature", "Returns the shutdown critical
    AH_PREDEF ("_CSD",    "C-State Dependencies", "Returns a list of C-state dep
    AH_PREDEF ("_CST",    "C-States", "Returns a list of supported C-states"),
    AH_PREDEF ("_CWS",    "Clear Wake Alarm Status", "Clear the status of wake a
    AH_PREDEF ("_DBT",    "Debounce Timeout", "Timeout value, Resource Descripto
    AH_PREDEF ("_DCK",    "Dock Present", "Sets docking isolation. Presence indi
    AH_PREDEF ("_DCS",    "Display Current Status", "Returns status of the displ
    AH_PREDEF ("_DDC",    "Display Data Current", "Returns the EDID for the disp
    AH_PREDEF ("_DDN",    "DOS Device Name", "Returns a device logical name"),
    AH_PREDEF ("_DEC",    "Decode", "Device decoding type, Resource Descriptor f
    AH_PREDEF ("_DEP",    "Dependencies", "Returns a list of operation region de
    AH_PREDEF ("_DGS",    "Display Graphics State", "Return the current state of
    AH_PREDEF ("_DIS",    "Disable Device", "Disables a device"),
    AH_PREDEF ("_DLM",    "Device Lock Mutex", "Defines mutex for OS/AML sharing
    AH_PREDEF ("_DMA",    "Direct Memory Access", "Returns device current resour
    AH_PREDEF ("_DOD",    "Display Output Devices", "Enumerate all devices attac
    AH_PREDEF ("_DOS",    "Disable Output Switching", "Sets the display output s
    AH_PREDEF ("_DPL",    "Device Selection Polarity", "Polarity of Device Selec
    AH_PREDEF ("_DRS",    "Drive Strength", "Drive Strength setting for GPIO con
    AH_PREDEF ("_DSM",    "Device-Specific Method", "Executes device-specific fu
    AH_PREDEF ("_DSS",    "Device Set State", "Sets the display device state"),
    AH_PREDEF ("_DSW",    "Device Sleep Wake", "Sets the sleep and wake transiti
    AH_PREDEF ("_DTI",    "Device Temperature Indication", "Conveys native devic
    AH_PREDEF ("_Exx",    "Edge-Triggered GPE", "Method executed as a result of
    AH_PREDEF ("_EC_",    "Embedded Controller", "returns EC offset and query in
    AH_PREDEF ("_EDL",    "Eject Device List", "Returns a list of devices that a
```

```
128     AH_PREDEF ("_EJD",    "Ejection Dependent Device", "Returns the name of depe
129     AH_PREDEF ("_EJx",    "Eject Device", "Begin or cancel a device ejection req
130     AH_PREDEF ("_END",    "Endianness", "Endian orientation, Resource Descriptor
131     AH_PREDEF ("_EVT",    "Event", "Event method for GPIO events"),
132     AH_PREDEF ("_FDE",    "Floppy Disk Enumerate", "Returns floppy disk configur
133     AH_PREDEF ("_FDI",    "Floppy Drive Information", "Returns a floppy drive in
134     AH_PREDEF ("_FDM",    "Floppy Drive Mode", "Sets a floppy drive speed"),
135     AH_PREDEF ("_FIF",    "Fan Information", "Returns fan device information"),
136     AH_PREDEF ("_FIX",    "Fixed Register Resource Provider", "Returns a list of
137     AH_PREDEF ("_FLC",    "Flow Control", "Flow control, Resource Descriptor fie
138     AH_PREDEF ("_FPS",    "Fan Performance States", "Returns a list of supported
139     AH_PREDEF ("_FSL",    "Fan Set Level", "Control method that sets the fan dev
140     AH_PREDEF ("_FST",    "Fan Status", "Returns current status information for
141     AH_PREDEF ("_GAI",    "Get Averaging Interval", "Returns the power meter ave
142     AH_PREDEF ("_GCP",    "Get Capabilities", "Get device time capabilities"),
143     AH_PREDEF ("_GHL",    "Get Hardware Limit", "Returns the hardware limit enfo
144     AH_PREDEF ("_GL_",    "Global Lock", "OS-defined Global Lock mutex object"),
145     AH_PREDEF ("_GLK",    "Get Global Lock Requirement", "Returns a device's Glo
146     AH_PREDEF ("_GPD",    "Get Post Data", "Returns the value of the VGA device
147     AH_PREDEF ("_GPE",    "General Purpose Events", "Predefined scope (\\_GPE) o
148     AH_PREDEF ("_GRA",    "Granularity", "Address space granularity, Resource De
149     AH_PREDEF ("_GRT",    "Get Real Time", "Returns current time-of-day from a t
150     AH_PREDEF ("_GSB",    "Global System Interrupt Base", "Returns the GSB for a
151     AH_PREDEF ("_GTF",    "Get Task File", "Returns a list of ATA commands to re
152     AH_PREDEF ("_GTM",    "Get Timing Mode", "Returns a list of IDE controller t
153     AH_PREDEF ("_GTS",    "Going To Sleep", "Inform AML of pending sleep"),
154     AH_PREDEF ("_GWS",    "Get Wake Status", "Return status of wake alarms"),
155     AH_PREDEF ("_HE_",    "High-Edge", "Interrupt triggering, Resource Descripto
156     AH_PREDEF ("_HID",    "Hardware ID", "Returns a device's Plug and Play Hardw
157     AH_PREDEF ("_HOT",    "Hot Temperature", "Returns the critical temperature f
158     AH_PREDEF ("_HPP",    "Hot Plug Parameters", "Returns a list of hot-plug inf
159     AH_PREDEF ("_HPX",    "Hot Plug Parameter Extensions", "Returns a list of ho
160     AH_PREDEF ("_HRV",    "Hardware Revision", "Returns a hardware revision valu
161     AH_PREDEF ("_IFT",    "IPMI Interface Type", "See the Intelligent Platform M
162     AH_PREDEF ("_INI",    "Initialize", "Performs device specific initialization
163     AH_PREDEF ("_INT",    "Interrupts", "Interrupt mask bits, Resource Descripto
164     AH_PREDEF ("_IOR",    "I/O Restriction", "Restriction type, Resource Descrip
165     AH_PREDEF ("_IRC",    "Inrush Current", "Presence indicates that a device ha
166     AH_PREDEF ("_Lxx",    "Level-Triggered GPE", "Control method executed as a r
167     AH_PREDEF ("_LCK",    "Lock Device", "Locks or unlocks a device (docking)"),
168     AH_PREDEF ("_LEN",    "Length", "Range length, Resource Descriptor field"),
169     AH_PREDEF ("_LID",    "Lid Status", "Returns the open/closed status of the l
170     AH_PREDEF ("_LIN",    "Lines In Use", "Handshake lines, Resource Descriptor
171     AH_PREDEF ("_LL_",    "Low Level", "Interrupt polarity, Resource Descriptor
172     AH_PREDEF ("_MAF",    "Maximum Address Fixed", "Resource Descriptor field"),
173     AH_PREDEF ("_MAT",    "Multiple APIC Table Entry", "Returns a list of MADT A
174     AH_PREDEF ("_MAX",    "Maximum Base Address", "Resource Descriptor field"),
175     AH_PREDEF ("_MBM",    "Memory Bandwidth Monitoring Data", "Returns bandwidth
176     AH_PREDEF ("_MEM",    "Memory Attributes", "Resource Descriptor field"),
177     AH_PREDEF ("_MIF",    "Minimum Address Fixed", "Resource Descriptor field"),
178     AH_PREDEF ("_MIN",    "Minimum Base Address", "Resource Descriptor field"),
179     AH_PREDEF ("_MLS",    "Multiple Language String", "Returns a device descript
180     AH_PREDEF ("_MOD",    "Mode", "Interrupt mode, Resource Descriptor field"),
181     AH_PREDEF ("_MSG",    "Message", "Sets the system message waiting status ind
182     AH_PREDEF ("_MSM",    "Memory Set Monitoring", "Sets bandwidth monitoring pa
183     AH_PREDEF ("_MTP",    "Memory Type", "Resource Descriptor field"),
184     AH_PREDEF ("_NTT",    "Notification Temperature Threshold", "Returns a thres
185     AH_PREDEF ("_OFF",    "Power Off", "Sets a power resource to the off state")
186     AH_PREDEF ("_ON_",    "Power On", "Sets a power resource to the on state"),
187     AH_PREDEF ("_OS_",    "Operating System", "Returns a string that identifies
188     AH_PREDEF ("_OSC",    "Operating System Capabilities", "Inform AML of host f
189     AH_PREDEF ("_OSI",    "Operating System Interfaces", "Returns supported inte
190     AH_PREDEF ("_OST",    "OSPM Status Indication", "Inform AML of event process
191     AH_PREDEF ("_PAI",    "Power Averaging Interval", "Sets the averaging interv
192     AH_PREDEF ("_PAR",    "Parity", "Parity bits, Resource Descriptor field"),
193     AH_PREDEF ("_PCL",    "Power Consumer List", "Returns a list of devices powe
```



```
194     AH_PREDEF ("_PCT",    "Performance Control", "Returns processor performance
195     AH_PREDEF ("_PDC",    "Processor Driver Capabilities", "Inform AML of proces
196     AH_PREDEF ("_PDL",    "P-state Depth Limit", "Returns the lowest available p
197     AH_PREDEF ("_PHA",    "Clock Phase", "Clock phase, Resource Descriptor field
198     AH_PREDEF ("_PIC",    "Interrupt Model", "Inform AML of the interrupt model
199     AH_PREDEF ("_PIF",    "Power Source Information", "Returns a Power Source in
200     AH_PREDEF ("_PIN",    "Pin List", "Pin list, Resource Descriptor field"),
201     AH_PREDEF ("_PLD",    "Physical Location of Device", "Returns a device's phy
202     AH_PREDEF ("_PMC",    "Power Meter Capabilities", "Returns a list of Power M
203     AH_PREDEF ("_PMD",    "Power Metered Devices", "Returns a list of devices th
204     AH_PREDEF ("_PMM",    "Power Meter Measurement", "Returns the current value
205     AH_PREDEF ("_POL",    "Polarity", "Interrupt polarity, Resource Descriptor f
206     AH_PREDEF ("_PPC",    "Performance Present Capabilites", "Returns a list of
207     AH_PREDEF ("_PPE",    "Polling for Platform Error", "Returns the polling int
208     AH_PREDEF ("_PPI",    "Pin Configuration", "Resource Descriptor field"),
209     AH_PREDEF ("_PR",     "Processor", "Predefined scope for processor objects")
210     AH_PREDEF ("_PR0",    "Power Resources for D0", "Returns a list of dependent
211     AH_PREDEF ("_PR1",    "Power Resources for D1", "Returns a list of dependent
212     AH_PREDEF ("_PR2",    "Power Resources for D2", "Returns a list of dependent
213     AH_PREDEF ("_PR3",    "Power Resources for D3hot", "Returns a list of depend
214     AH_PREDEF ("_PRE",    "Power Resources for Enumeration", "Returns a list of
215     AH_PREDEF ("_PRL",    "Power Source Redundancy List", "Returns a list of pow
216     AH_PREDEF ("_PRS",    "Possible Resource Settings", "Returns a list of a dev
217     AH_PREDEF ("_PRT",    "PCI Routing Table", "Returns a list of PCI interrupt
218     AH_PREDEF ("_PRW",    "Power Resources for Wake", "Returns a list of depende
219     AH_PREDEF ("_PS0",    "Power State 0", "Sets a device's power state to D0 (d
220     AH_PREDEF ("_PS1",    "Power State 1", "Sets a device's power state to D1"),
221     AH_PREDEF ("_PS2",    "Power State 2", "Sets a device's power state to D2"),
222     AH_PREDEF ("_PS3",    "Power State 3", "Sets a device's power state to D3 (d
223     AH_PREDEF ("_PSC",    "Power State Current", "Returns a device's current pow
224     AH_PREDEF ("_PSD",    "Power State Dependencies", "Returns processor P-State
225     AH_PREDEF ("_PSE",    "Power State for Enumeration", "Put a bus into enumera
226     AH_PREDEF ("_PSL",    "Passive List", "Returns a list of passive cooling dev
227     AH_PREDEF ("_PSR",    "Power Source", "Returns the power source device curre
228     AH_PREDEF ("_PSS",    "Performance Supported States", "Returns a list of sup
229     AH_PREDEF ("_PSV",    "Passive Temperature", "Returns the passive trip point
230     AH_PREDEF ("_PSW",    "Power State Wake", "Sets a device's wake function"),
231     AH_PREDEF ("_PTC",    "Processor Throttling Control", "Returns throttling co
232     AH_PREDEF ("_PTP",    "Power Trip Points", "Sets trip points for the Power M
233     AH_PREDEF ("_PTS",    "Prepare To Sleep", "Inform the platform of an impendi
234     AH_PREDEF ("_PUR",    "Processor Utilization Request", "Returns the number o
235     AH_PREDEF ("_PXM",    "Device Proximity", "Returns a device's proximity doma
236     AH_PREDEF ("_Qxx",    "EC Query", "Embedded Controller query and SMBus Alarm
237     AH_PREDEF ("_RBO",    "Register Bit Offset", "Resource Descriptor field"),
238     AH_PREDEF ("_RBW",    "Register Bit Width", "Resource Descriptor field"),
239     AH_PREDEF ("_REG",    "Region Availability", "Inform AML code of an operatio
240     AH_PREDEF ("_REV",    "Supported ACPI Revision", "Returns the revision of th
241     AH_PREDEF ("_RMV",    "Removal Status", "Returns a device's removal ability
242     AH_PREDEF ("_RNG",    "Range", "Memory range type, Resource Descriptor field
243     AH_PREDEF ("_ROM",    "Read-Only Memory", "Returns a copy of the ROM data fo
244     AH_PREDEF ("_RT_",    "Resource Type", "Resource Descriptor field"),
245     AH_PREDEF ("_RTV",    "Relative Temperature Values", "Returns temperature va
246     AH_PREDEF ("_RW_",    "Read-Write Status", "Resource Descriptor field"),
247     AH_PREDEF ("_RXL",    "Receive Buffer Size", "Serial channel buffer, Resourc
248     AH_PREDEF ("_S0_",    "S0 System State", "Returns values to enter the system
249     AH_PREDEF ("_S1_",    "S1 System State", "Returns values to enter the system
250     AH_PREDEF ("_S2_",    "S2 System State", "Returns values to enter the system
251     AH_PREDEF ("_S3_",    "S3 System State", "Returns values to enter the system
252     AH_PREDEF ("_S4_",    "S4 System State", "Returns values to enter the system
253     AH_PREDEF ("_S5_",    "S5 System State", "Returns values to enter the system
254     AH_PREDEF ("_S1D",    "S1 Device State", "Returns the highest D-state suppor
255     AH_PREDEF ("_S2D",    "S2 Device State", "Returns the highest D-state suppor
256     AH_PREDEF ("_S3D",    "S3 Device State", "Returns the highest D-state suppor
257     AH_PREDEF ("_S4D",    "S4 Device State", "Returns the highest D-state suppor
258     AH_PREDEF ("_S0W",    "S0 Device Wake State", "Returns the lowest D-state th
259     AH_PREDEF ("_S1W",    "S1 Device Wake State", "Returns the lowest D-state fo
```

```
260     AH_PREDEF ("_S2W",    "S2 Device Wake State", "Returns the lowest D-state fo
261     AH_PREDEF ("_S3W",    "S3 Device Wake State", "Returns the lowest D-state fo
262     AH_PREDEF ("_S4W",    "S4 Device Wake State", "Returns the lowest D-state fo
263     AH_PREDEF ("_SB_",    "System Bus", "Predefined scope for device and bus obj
264     AH_PREDEF ("_SBS",    "Smart Battery Subsystem", "Returns the subsystem conf
265     AH_PREDEF ("_SCP",    "Set Cooling Policy", "Sets the cooling policy (active
266     AH_PREDEF ("_SDD",    "Set Device Data", "Sets data for a SATA device"),
267     AH_PREDEF ("_SEG",    "PCI Segment", "Returns a device's PCI Segment Group n
268     AH_PREDEF ("_SHL",    "Set Hardware Limit", "Sets the hardware limit enforce
269     AH_PREDEF ("_SHR",    "Sharable", "Interrupt share status, Resource Descript
270     AH_PREDEF ("_SI_",    "System Indicators", "Predefined scope"),
271     AH_PREDEF ("_SIZ",    "Size", "DMA transfer size, Resource Descriptor field"
272     AH_PREDEF ("_SLI",    "System Locality Information", "Returns a list of NUMA
273     AH_PREDEF ("_SLV",    "Slave Mode", "Mode setting, Resource Descriptor field
274     AH_PREDEF ("_SPD",    "Set Post Device", "Sets which video device will be po
275     AH_PREDEF ("_SPE",    "Speed", "Connection speed, Resource Descriptor field"
276     AH_PREDEF ("_SRS",    "Set Resource Settings", "Sets a device's resource all
277     AH_PREDEF ("_SRT",    "Set Real Time", "Sets the current time for a time/ala
278     AH_PREDEF ("_SRV",    "IPMI Spec Revision", "See the Intelligent Platform Ma
279     AH_PREDEF ("_SST",    "System Status", "Sets the system status indicator"),
280     AH_PREDEF ("_STA",    "Status", "Returns the current status of a Device or P
281     AH_PREDEF ("_STB",    "Stop Bits", "Serial channel stop bits, Resource Descr
282     AH_PREDEF ("_STM",    "Set Timing Mode", "Sets an IDE controller transfer ti
283     AH_PREDEF ("_STP",    "Set Expired Timer Wake Policy", "Sets expired timer p
284     AH_PREDEF ("_STR",    "Description String", "Returns a device's description
285     AH_PREDEF ("_STV",    "Set Timer Value", "Set timer values of the wake alarm
286     AH_PREDEF ("_SUB",    "Subsystem ID", "Returns the subsystem ID for a device
287     AH_PREDEF ("_SUN",    "Slot User Number", "Returns the slot unique ID number
288     AH_PREDEF ("_SWS",    "System Wake Source", "Returns the source event that c
289     AH_PREDEF ("_T_x",    "Emitted by ASL Compiler", "Reserved for use by ASL co
290     AH_PREDEF ("_TC1",    "Thermal Constant 1", "Returns TC1 for the passive coo
291     AH_PREDEF ("_TC2",    "Thermal Constant 2", "Returns TC2 for the passive coo
292     AH_PREDEF ("_TDL",    "T-State Depth Limit", "Returns the _TSS entry number
293     AH_PREDEF ("_TIP",    "Expired Timer Wake Policy", "Returns timer policies o
294     AH_PREDEF ("_TIV",    "Timer Values", "Returns remaining time of the wake al
295     AH_PREDEF ("_TMP",    "Temperature", "Returns a thermal zone's current tempe
296     AH_PREDEF ("_TPC",    "Throttling Present Capabilities", "Returns the curren
297     AH_PREDEF ("_TPT",    "Trip Point Temperature", "Inform AML that a device's
298     AH_PREDEF ("_TRA",    "Translation", "Address translation offset, Resource D
299     AH_PREDEF ("_TRS",    "Translation Sparse", "Sparse/dense flag, Resource Des
300     AH_PREDEF ("_TRT",    "Thermal Relationship Table", "Returns thermal relatio
301     AH_PREDEF ("_TSD",    "Throttling State Dependencies", "Returns a list of T-
302     AH_PREDEF ("_TSF",    "Type-Specific Flags", "Resource Descriptor field"),
303     AH_PREDEF ("_TSP",    "Thermal Sampling Period", "Returns the thermal sampli
304     AH_PREDEF ("_TSS",    "Throttling Supported States", "Returns supported thro
305     AH_PREDEF ("_TST",    "Temperature Sensor Threshold", "Returns the minimum s
306     AH_PREDEF ("_TTP",    "Translation Type", "Translation/static flag, Resource
307     AH_PREDEF ("_TTS",    "Transition To State", "Inform AML of an S-state trans
308     AH_PREDEF ("_TXL",    "Transmit Buffer Size", "Serial Channel buffer, Resour
309     AH_PREDEF ("_TYP",    "Type", "DMA channel type (speed), Resource Descriptor
310     AH_PREDEF ("_TZ_",    "Thermal Zone", "Predefined scope: ACPI 1.0"),
311     AH_PREDEF ("_TZD",    "Thermal Zone Devices", "Returns a list of device name
312     AH_PREDEF ("_TZM",    "Thermal Zone Member", "Returns a reference to the the
313     AH_PREDEF ("_TZP",    "Thermal Zone Polling", "Returns a Thermal zone's poll
314     AH_PREDEF ("_UID",    "Unique ID", "Return a device's unique persistent ID")
315     AH_PREDEF ("_UPC",    "USB Port Capabilities", "Returns a list of USB port c
316     AH_PREDEF ("_UPD",    "User Presence Detect", "Returns user detection inform
317     AH_PREDEF ("_UPP",    "User Presence Polling", "Returns the recommended user
318     AH_PREDEF ("_VEN",    "Vendor Data", "Resource Descriptor field"),
319     AH_PREDEF ("_VPO",    "Video Post Options", "Returns the implemented video p
320     AH_PREDEF ("_WAK",    "Wake", "Inform AML that the system has just awakened"
321     AH_PREDEF ("_Wxx",    "Wake Event", "Method executed as a result of a wake e
322     AH_PREDEF (NULL,      NULL, NULL)
323 };
```

```
**********************************************************
   38442 Thu Dec 26 13:48:23 2013
new/usr/src/common/acpica/common/dmextern.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmextern - Support for External() ASL statements
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acnamesp.h"
#include "acdisasm.h"
#include "aslcompiler.h"
#include <stdio.h>
#include <errno.h>


/*
 * This module is used for application-level code (iASL disassembler) only.
 *
 * It contains the code to create and emit any necessary External() ASL
 * statements for the module being disassembled.
```



```
 */
#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmextern")


/*
 * This table maps ACPI_OBJECT_TYPEs to the corresponding ASL
 * ObjectTypeKeyword. Used to generate typed external declarations
 */
static const char           *AcpiGbl_DmTypeNames[] =
{
    /* 00 */ "",                    /* Type ANY */
    /* 01 */ ", IntObj",
    /* 02 */ ", StrObj",
    /* 03 */ ", BuffObj",
    /* 04 */ ", PkgObj",
    /* 05 */ ", FieldUnitObj",
    /* 06 */ ", DeviceObj",
    /* 07 */ ", EventObj",
    /* 08 */ ", MethodObj",
    /* 09 */ ", MutexObj",
    /* 10 */ ", OpRegionObj",
    /* 11 */ ", PowerResObj",
    /* 12 */ ", ProcessorObj",
    /* 13 */ ", ThermalZoneObj",
    /* 14 */ ", BuffFieldObj",
    /* 15 */ ", DDBHandleObj",
    /* 16 */ "",                    /* Debug object */
    /* 17 */ ", FieldUnitObj",
    /* 18 */ ", FieldUnitObj",
    /* 19 */ ", FieldUnitObj"
};

#define METHOD_SEPARATORS           " \t,()\n"


/* Local prototypes */

static const char *
AcpiDmGetObjectTypeName (
    ACPI_OBJECT_TYPE        Type);

static char *
AcpiDmNormalizeParentPrefix (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Path);

static void
AcpiDmAddPathToExternalList (
    char                    *Path,
    UINT8                   Type,
    UINT32                  Value,
    UINT16                  Flags);

static ACPI_STATUS
AcpiDmCreateNewExternal (
    char                    *ExternalPath,
    char                    *InternalPath,
    UINT8                   Type,
    UINT32                  Value,
    UINT16                  Flags);


/******************************************************************************
 *
 * FUNCTION:    AcpiDmGetObjectTypeName
```

```
 *
 * PARAMETERS:  Type                - An ACPI_OBJECT_TYPE
 *
 * RETURN:      Pointer to a string
 *
 * DESCRIPTION: Map an object type to the ASL object type string.
 *
 ******************************************************************************/

static const char *
AcpiDmGetObjectTypeName (
    ACPI_OBJECT_TYPE        Type)
{

    if (Type == ACPI_TYPE_LOCAL_SCOPE)
    {
        Type = ACPI_TYPE_DEVICE;
    }

    else if (Type > ACPI_TYPE_LOCAL_INDEX_FIELD)
    {
        return ("");
    }

    return (AcpiGbl_DmTypeNames[Type]);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmNormalizeParentPrefix
 *
 * PARAMETERS:  Op                  - Parse op
 *              Path                - Path with parent prefix
 *
 * RETURN:      The full pathname to the object (from the namespace root)
 *
 * DESCRIPTION: Returns the full pathname of a path with parent prefix
 *              The caller must free the fullpath returned.
 *
 ******************************************************************************/

static char *
AcpiDmNormalizeParentPrefix (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Path)
{
    ACPI_NAMESPACE_NODE     *Node;
    char                    *Fullpath;
    char                    *ParentPath;
    ACPI_SIZE               Length;
    UINT32                  Index = 0;


    if (!Op)
    {
        return (NULL);
    }

    /* Search upwards in the parse tree until we reach the next namespace node *

    Op = Op->Common.Parent;
    while (Op)
    {
        if (Op->Common.Node)
        {
```



```
            break;
        }

        Op = Op->Common.Parent;
    }

    if (!Op)
    {
        return (NULL);
    }

    /*
     * Find the actual parent node for the reference:
     * Remove all carat prefixes from the input path.
     * There may be multiple parent prefixes (For example, ^^^M000)
     */
    Node = Op->Common.Node;
    while (Node && (*Path == (UINT8) AML_PARENT_PREFIX))
    {
        Node = Node->Parent;
        Path++;
    }

    if (!Node)
    {
        return (NULL);
    }

    /* Get the full pathname for the parent node */

    ParentPath = AcpiNsGetExternalPathname (Node);
    if (!ParentPath)
    {
        return (NULL);
    }

    Length = (ACPI_STRLEN (ParentPath) + ACPI_STRLEN (Path) + 1);
    if (ParentPath[1])
    {
        /*
         * If ParentPath is not just a simple '\', increment the length
         * for the required dot separator (ParentPath.Path)
         */
        Length++;

        /* For External() statements, we do not want a leading '\' */

        if (*ParentPath == AML_ROOT_PREFIX)
        {
            Index = 1;
        }
    }

    Fullpath = ACPI_ALLOCATE_ZEROED (Length);
    if (!Fullpath)
    {
        goto Cleanup;
    }

    /*
     * Concatenate parent fullpath and path. For example,
     * parent fullpath "\_SB_", Path "^INIT", Fullpath "\_SB_.INIT"
     *
     * Copy the parent path
     */
    ACPI_STRCPY (Fullpath, &ParentPath[Index]);
```

```
258     /*
259      * Add dot separator
260      * (don't need dot if parent fullpath is a single backslash)
261      */
262     if (ParentPath[1])
263     {
264         ACPI_STRCAT (Fullpath, ".");
265     }

267     /* Copy child path (carat parent prefix(es) were skipped above) */

269     ACPI_STRCAT (Fullpath, Path);

271 Cleanup:
272     ACPI_FREE (ParentPath);
273     return (Fullpath);
274 }


277 /*******************************************************************************
278  *
279  * FUNCTION:    AcpiDmAddToExternalFileList
280  *
281  * PARAMETERS:  PathList            - Single path or list separated by comma
282  *
283  * RETURN:      None
284  *
285  * DESCRIPTION: Add external files to global list
286  *
287  ******************************************************************************/

289 ACPI_STATUS
290 AcpiDmAddToExternalFileList (
291     char                    *Pathname)
292 {
293     ACPI_EXTERNAL_FILE      *ExternalFile;
294     char                    *LocalPathname;


297     if (!Pathname)
298     {
299         return (AE_OK);
300     }

302     LocalPathname = ACPI_ALLOCATE (strlen (Pathname) + 1);
303     if (!LocalPathname)
304     {
305         return (AE_NO_MEMORY);
306     }

308     ExternalFile = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EXTERNAL_FILE));
309     if (!ExternalFile)
310     {
311         ACPI_FREE (LocalPathname);
312         return (AE_NO_MEMORY);
313     }

315     /* Take a copy of the file pathname */

317     strcpy (LocalPathname, Pathname);
318     ExternalFile->Path = LocalPathname;

320     if (AcpiGbl_ExternalFileList)
321     {
322         ExternalFile->Next = AcpiGbl_ExternalFileList;
```



```
323     }

325     AcpiGbl_ExternalFileList = ExternalFile;
326     return (AE_OK);
327 }


330 /*******************************************************************************
331  *
332  * FUNCTION:    AcpiDmClearExternalFileList
333  *
334  * PARAMETERS:  None
335  *
336  * RETURN:      None
337  *
338  * DESCRIPTION: Clear the external file list
339  *
340  ******************************************************************************/

342 void
343 AcpiDmClearExternalFileList (
344     void)
345 {
346     ACPI_EXTERNAL_FILE      *NextExternal;


349     while (AcpiGbl_ExternalFileList)
350     {
351         NextExternal = AcpiGbl_ExternalFileList->Next;
352         ACPI_FREE (AcpiGbl_ExternalFileList->Path);
353         ACPI_FREE (AcpiGbl_ExternalFileList);
354         AcpiGbl_ExternalFileList = NextExternal;
355     }
356 }


359 /*******************************************************************************
360  *
361  * FUNCTION:    AcpiDmGetExternalsFromFile
362  *
363  * PARAMETERS:  None
364  *
365  * RETURN:      None
366  *
367  * DESCRIPTION: Process the optional external reference file.
368  *
369  * Each line in the file should be of the form:
370  *      External (<Method namepath>, MethodObj, <ArgCount>)
371  *
372  * Example:
373  *      External (_SB_.PCI0.XHC_.PS0X, MethodObj, 4)
374  *
375  ******************************************************************************/

377 void
378 AcpiDmGetExternalsFromFile (
379     void)
380 {
381     FILE                    *ExternalRefFile;
382     char                    *Token;
383     char                    *MethodName;
384     UINT32                  ArgCount;
385     UINT32                  ImportCount = 0;


388     if (!Gbl_ExternalRefFilename)
```

```
389     {
390         return;
391     }

393     /* Open the file */

395     ExternalRefFile = fopen (Gbl_ExternalRefFilename, "r");
396     if (!ExternalRefFile)
397     {
398         fprintf (stderr, "Could not open external reference file \"%s\"\n",
399             Gbl_ExternalRefFilename);
400         return;
401     }

403     /* Each line defines a method */

405     while (fgets (StringBuffer, ASL_MSG_BUFFER_SIZE, ExternalRefFile))
406     {
407         Token = strtok (StringBuffer, METHOD_SEPARATORS);   /* "External" */
408         if (!Token) continue;
409         if (strcmp (Token, "External")) continue;

411         MethodName = strtok (NULL, METHOD_SEPARATORS);      /* Method namepath *
412         if (!MethodName) continue;

414         Token = strtok (NULL, METHOD_SEPARATORS);           /* "MethodObj" */
415         if (!Token) continue;
416         if (strcmp (Token, "MethodObj")) continue;

418         Token = strtok (NULL, METHOD_SEPARATORS);           /* Arg count */
419         if (!Token) continue;

421         /* Convert arg count string to an integer */

423         errno = 0;
424         ArgCount = strtoul (Token, NULL, 0);
425         if (errno)
426         {
427             fprintf (stderr, "Invalid argument count (%s)\n", Token);
428             continue;
429         }
430         if (ArgCount > 7)
431         {
432             fprintf (stderr, "Invalid argument count (%u)\n", ArgCount);
433             continue;
434         }

436         /* Add this external to the global list */

438         AcpiOsPrintf ("%s: Importing method external (%u arguments) %s\n",
439             Gbl_ExternalRefFilename, ArgCount, MethodName);

441         AcpiDmAddPathToExternalList (MethodName, ACPI_TYPE_METHOD,
442             ArgCount, (ACPI_EXT_RESOLVED_REFERENCE | ACPI_EXT_ORIGIN_FROM_FILE))
443         ImportCount++;
444     }

446     if (!ImportCount)
447     {
448         fprintf (stderr, "Did not find any external methods in reference file \"
449             Gbl_ExternalRefFilename);
450     }
451     else
452     {
453         /* Add the external(s) to the namespace */
```



```
455         AcpiDmAddExternalsToNamespace ();

457         AcpiOsPrintf ("%s: Imported %u external method definitions\n",
458             Gbl_ExternalRefFilename, ImportCount);
459     }

461     fclose (ExternalRefFile);
462 }


465 /*******************************************************************************
466  *
467  * FUNCTION:    AcpiDmAddOpToExternalList
468  *
469  * PARAMETERS:  Op                  - Current parser Op
470  *              Path                - Internal (AML) path to the object
471  *              Type                - ACPI object type to be added
472  *              Value               - Arg count if adding a Method object
473  *              Flags               - To be passed to the external object
474  *
475  * RETURN:      None
476  *
477  * DESCRIPTION: Insert a new name into the global list of Externals which
478  *              will in turn be later emitted as an External() declaration
479  *              in the disassembled output.
480  *
481  *              This function handles the most common case where the referenced
482  *              name is simply not found in the constructed namespace.
483  *
484  ******************************************************************************/

486 void
487 AcpiDmAddOpToExternalList (
488     ACPI_PARSE_OBJECT       *Op,
489     char                    *Path,
490     UINT8                   Type,
491     UINT32                  Value,
492     UINT16                  Flags)
493 {
494     char                    *ExternalPath;
495     char                    *InternalPath = Path;
496     char                    *Temp;
497     ACPI_STATUS             Status;


500     ACPI_FUNCTION_TRACE (DmAddOpToExternalList);


503     if (!Path)
504     {
505         return_VOID;
506     }

508     /* Remove a root backslash if present */

510     if ((*Path == AML_ROOT_PREFIX) && (Path[1]))
511     {
512         Path++;
513     }

515     /* Externalize the pathname */

517     Status = AcpiNsExternalizeName (ACPI_UINT32_MAX, Path,
518         NULL, &ExternalPath);
519     if (ACPI_FAILURE (Status))
520     {
```

```
521         return_VOID;
522     }

524     /*
525      * Get the full pathname from the root if "Path" has one or more
526      * parent prefixes (^). Note: path will not contain a leading '\'.
527      */
528     if (*Path == (UINT8) AML_PARENT_PREFIX)
529     {
530         Temp = AcpiDmNormalizeParentPrefix (Op, ExternalPath);

532         /* Set new external path */

534         ACPI_FREE (ExternalPath);
535         ExternalPath = Temp;
536         if (!Temp)
537         {
538             return_VOID;
539         }

541         /* Create the new internal pathname */

543         Flags |= ACPI_EXT_INTERNAL_PATH_ALLOCATED;
544         Status = AcpiNsInternalizeName (ExternalPath, &InternalPath);
545         if (ACPI_FAILURE (Status))
546         {
547             ACPI_FREE (ExternalPath);
548             return_VOID;
549         }
550     }

552     /* Create the new External() declaration node */

554     Status = AcpiDmCreateNewExternal (ExternalPath, InternalPath,
555         Type, Value, Flags);
556     if (ACPI_FAILURE (Status))
557     {
558         ACPI_FREE (ExternalPath);
559         if (Flags & ACPI_EXT_INTERNAL_PATH_ALLOCATED)
560         {
561             ACPI_FREE (InternalPath);
562         }
563     }

565     return_VOID;
566 }


569 /*******************************************************************************
570  *
571  * FUNCTION:    AcpiDmAddNodeToExternalList
572  *
573  * PARAMETERS:  Node                - Namespace node for object to be added
574  *              Type                - ACPI object type to be added
575  *              Value               - Arg count if adding a Method object
576  *              Flags               - To be passed to the external object
577  *
578  * RETURN:      None
579  *
580  * DESCRIPTION: Insert a new name into the global list of Externals which
581  *              will in turn be later emitted as an External() declaration
582  *              in the disassembled output.
583  *
584  *              This function handles the case where the referenced name has
585  *              been found in the namespace, but the name originated in a
586  *              table other than the one that is being disassembled (such
```



```
587  *              as a table that is added via the iASL -e option).
588  *
589  ******************************************************************************/

591 void
592 AcpiDmAddNodeToExternalList (
593     ACPI_NAMESPACE_NODE     *Node,
594     UINT8                   Type,
595     UINT32                  Value,
596     UINT16                  Flags)
597 {
598     char                    *ExternalPath;
599     char                    *InternalPath;
600     char                    *Temp;
601     ACPI_STATUS             Status;


604     ACPI_FUNCTION_TRACE (DmAddNodeToExternalList);


607     if (!Node)
608     {
609         return_VOID;
610     }

612     /* Get the full external and internal pathnames to the node */

614     ExternalPath = AcpiNsGetExternalPathname (Node);
615     if (!ExternalPath)
616     {
617         return_VOID;
618     }

620     Status = AcpiNsInternalizeName (ExternalPath, &InternalPath);
621     if (ACPI_FAILURE (Status))
622     {
623         ACPI_FREE (ExternalPath);
624         return_VOID;
625     }

627     /* Remove the root backslash */

629     if ((*ExternalPath == AML_ROOT_PREFIX) && (ExternalPath[1]))
630     {
631         Temp = ACPI_ALLOCATE_ZEROED (ACPI_STRLEN (ExternalPath) + 1);
632         if (!Temp)
633         {
634             return_VOID;
635         }

637         ACPI_STRCPY (Temp, &ExternalPath[1]);
638         ACPI_FREE (ExternalPath);
639         ExternalPath = Temp;
640     }

642     /* Create the new External() declaration node */

644     Status = AcpiDmCreateNewExternal (ExternalPath, InternalPath, Type,
645         Value, (Flags | ACPI_EXT_INTERNAL_PATH_ALLOCATED));
646     if (ACPI_FAILURE (Status))
647     {
648         ACPI_FREE (ExternalPath);
649         ACPI_FREE (InternalPath);
650     }

652     return_VOID;
```

```
653 }


656 /*******************************************************************************
657  *
658  * FUNCTION:    AcpiDmAddPathToExternalList
659  *
660  * PARAMETERS:  Path                - External name of the object to be added
661  *              Type                - ACPI object type to be added
662  *              Value               - Arg count if adding a Method object
663  *              Flags               - To be passed to the external object
664  *
665  * RETURN:      None
666  *
667  * DESCRIPTION: Insert a new name into the global list of Externals which
668  *              will in turn be later emitted as an External() declaration
669  *              in the disassembled output.
670  *
671  *              This function currently is used to add externals via a
672  *              reference file (via the -fe iASL option).
673  *
674  ******************************************************************************/

676 static void
677 AcpiDmAddPathToExternalList (
678     char                    *Path,
679     UINT8                   Type,
680     UINT32                  Value,
681     UINT16                  Flags)
682 {
683     char                    *InternalPath;
684     char                    *ExternalPath;
685     ACPI_STATUS             Status;


688     ACPI_FUNCTION_TRACE (DmAddPathToExternalList);


691     if (!Path)
692     {
693         return_VOID;
694     }

696     /* Remove a root backslash if present */

698     if ((*Path == AML_ROOT_PREFIX) && (Path[1]))
699     {
700         Path++;
701     }

703     /* Create the internal and external pathnames */

705     Status = AcpiNsInternalizeName (Path, &InternalPath);
706     if (ACPI_FAILURE (Status))
707     {
708         return_VOID;
709     }

711     Status = AcpiNsExternalizeName (ACPI_UINT32_MAX, InternalPath,
712         NULL, &ExternalPath);
713     if (ACPI_FAILURE (Status))
714     {
715         ACPI_FREE (InternalPath);
716         return_VOID;
717     }
```



```
719     /* Create the new External() declaration node */

721     Status = AcpiDmCreateNewExternal (ExternalPath, InternalPath,
722         Type, Value, (Flags | ACPI_EXT_INTERNAL_PATH_ALLOCATED));
723     if (ACPI_FAILURE (Status))
724     {
725         ACPI_FREE (ExternalPath);
726         ACPI_FREE (InternalPath);
727     }

729     return_VOID;
730 }


733 /*******************************************************************************
734  *
735  * FUNCTION:    AcpiDmCreateNewExternal
736  *
737  * PARAMETERS:  ExternalPath        - External path to the object
738  *              InternalPath        - Internal (AML) path to the object
739  *              Type                - ACPI object type to be added
740  *              Value               - Arg count if adding a Method object
741  *              Flags               - To be passed to the external object
742  *
743  * RETURN:      Status
744  *
745  * DESCRIPTION: Common low-level function to insert a new name into the global
746  *              list of Externals which will in turn be later emitted as
747  *              External() declarations in the disassembled output.
748  *
749  *              Note: The external name should not include a root prefix
750  *              (backslash). We do not want External() statements to contain
751  *              a leading '\', as this prevents duplicate external statements
752  *              of the form:
753  *
754  *                  External (\ABCD)
755  *                  External (ABCD)
756  *
757  *              This would cause a compile time error when the disassembled
758  *              output file is recompiled.
759  *
760  *              There are two cases that are handled here. For both, we emit
761  *              an External() statement:
762  *              1) The name was simply not found in the namespace.
763  *              2) The name was found, but it originated in a table other than
764  *              the table that is being disassembled.
765  *
766  ******************************************************************************/

768 static ACPI_STATUS
769 AcpiDmCreateNewExternal (
770     char                    *ExternalPath,
771     char                    *InternalPath,
772     UINT8                   Type,
773     UINT32                  Value,
774     UINT16                  Flags)
775 {
776     ACPI_EXTERNAL_LIST      *NewExternal;
777     ACPI_EXTERNAL_LIST      *NextExternal;
778     ACPI_EXTERNAL_LIST      *PrevExternal = NULL;


781     ACPI_FUNCTION_TRACE (DmCreateNewExternal);


784     /* Check all existing externals to ensure no duplicates */
```

```
786     NextExternal = AcpiGbl_ExternalList;
787     while (NextExternal)
788     {
789         if (!ACPI_STRCMP (ExternalPath, NextExternal->Path))
790         {
791             /* Duplicate method, check that the Value (ArgCount) is the same */

793             if ((NextExternal->Type == ACPI_TYPE_METHOD) &&
794                 (NextExternal->Value != Value))
795             {
796                 ACPI_ERROR ((AE_INFO,
797                     "External method arg count mismatch %s: Current %u, attempte
798                     NextExternal->Path, NextExternal->Value, Value));
799             }

801             /* Allow upgrade of type from ANY */

803             else if (NextExternal->Type == ACPI_TYPE_ANY)
804             {
805                 NextExternal->Type = Type;
806                 NextExternal->Value = Value;
807             }

809             return_ACPI_STATUS (AE_ALREADY_EXISTS);
810         }

812         NextExternal = NextExternal->Next;
813     }

815     /* Allocate and init a new External() descriptor */

817     NewExternal = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EXTERNAL_LIST));
818     if (!NewExternal)
819     {
820         return_ACPI_STATUS (AE_NO_MEMORY);
821     }

823     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
824         "Adding external reference node (%s) type [%s]\n",
825         ExternalPath, AcpiUtGetTypeName (Type)));

827     NewExternal->Flags = Flags;
828     NewExternal->Value = Value;
829     NewExternal->Path = ExternalPath;
830     NewExternal->Type = Type;
831     NewExternal->Length = (UINT16) ACPI_STRLEN (ExternalPath);
832     NewExternal->InternalPath = InternalPath;

834     /* Link the new descriptor into the global list, alphabetically ordered */

836     NextExternal = AcpiGbl_ExternalList;
837     while (NextExternal)
838     {
839         if (AcpiUtStricmp (NewExternal->Path, NextExternal->Path) < 0)
840         {
841             if (PrevExternal)
842             {
843                 PrevExternal->Next = NewExternal;
844             }
845             else
846             {
847                 AcpiGbl_ExternalList = NewExternal;
848             }

850             NewExternal->Next = NextExternal;
```



```
851             return_ACPI_STATUS (AE_OK);
852         }

854         PrevExternal = NextExternal;
855         NextExternal = NextExternal->Next;
856     }

858     if (PrevExternal)
859     {
860         PrevExternal->Next = NewExternal;
861     }
862     else
863     {
864         AcpiGbl_ExternalList = NewExternal;
865     }

867     return_ACPI_STATUS (AE_OK);
868 }


871 /*******************************************************************************
872  *
873  * FUNCTION:    AcpiDmAddExternalsToNamespace
874  *
875  * PARAMETERS:  None
876  *
877  * RETURN:      None
878  *
879  * DESCRIPTION: Add all externals to the namespace. Allows externals to be
880  *              "resolved".
881  *
882  ******************************************************************************/

884 void
885 AcpiDmAddExternalsToNamespace (
886     void)
887 {
888     ACPI_STATUS             Status;
889     ACPI_NAMESPACE_NODE     *Node;
890     ACPI_OPERAND_OBJECT     *ObjDesc;
891     ACPI_EXTERNAL_LIST      *External = AcpiGbl_ExternalList;


894     while (External)
895     {
896         /* Add the external name (object) into the namespace */

898         Status = AcpiNsLookup (NULL, External->InternalPath, External->Type,
899                     ACPI_IMODE_LOAD_PASS1,
900                     ACPI_NS_ERROR_IF_FOUND | ACPI_NS_EXTERNAL | ACPI_NS_DONT_OPEN
901                     NULL, &Node);

903         if (ACPI_FAILURE (Status))
904         {
905             ACPI_EXCEPTION ((AE_INFO, Status,
906                 "while adding external to namespace [%s]",
907                 External->Path));
908         }

910         else switch (External->Type)
911         {
912         case ACPI_TYPE_METHOD:

914             /* For methods, we need to save the argument count */

916             ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_METHOD);
```

```
            ObjDesc->Method.ParamCount = (UINT8) External->Value;
            Node->Object = ObjDesc;
            break;

        case ACPI_TYPE_REGION:

            /* Regions require a region sub-object */

            ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_REGION);
            ObjDesc->Region.Node = Node;
            Node->Object = ObjDesc;
            break;

        default:

            break;
        }

        External = External->Next;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmGetExternalMethodCount
 *
 * PARAMETERS:  None
 *
 * RETURN:      The number of control method externals in the external list
 *
 * DESCRIPTION: Return the number of method externals that have been generated.
 *              If any control method externals have been found, we must
 *              re-parse the entire definition block with the new information
 *              (number of arguments for the methods.) This is limitation of
 *              AML, we don't know the number of arguments from the control
 *              method invocation itself.
 *
 ******************************************************************************/

UINT32
AcpiDmGetExternalMethodCount (
    void)
{
    ACPI_EXTERNAL_LIST      *External = AcpiGbl_ExternalList;
    UINT32                  Count = 0;


    while (External)
    {
        if (External->Type == ACPI_TYPE_METHOD)
        {
            Count++;
        }

        External = External->Next;
    }

    return (Count);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmClearExternalList
 *
```



```
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Free the entire External info list
 *
 ******************************************************************************/

void
AcpiDmClearExternalList (
    void)
{
    ACPI_EXTERNAL_LIST      *NextExternal;


    while (AcpiGbl_ExternalList)
    {
        NextExternal = AcpiGbl_ExternalList->Next;
        ACPI_FREE (AcpiGbl_ExternalList->Path);
        ACPI_FREE (AcpiGbl_ExternalList);
        AcpiGbl_ExternalList = NextExternal;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmEmitExternals
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Emit an External() ASL statement for each of the externals in
 *              the global external info list.
 *
 ******************************************************************************/

void
AcpiDmEmitExternals (
    void)
{
    ACPI_EXTERNAL_LIST      *NextExternal;


    if (!AcpiGbl_ExternalList)
    {
        return;
    }

    /*
     * Determine the number of control methods in the external list, and
     * also how many of those externals were resolved via the namespace.
     */
    NextExternal = AcpiGbl_ExternalList;
    while (NextExternal)
    {
        if (NextExternal->Type == ACPI_TYPE_METHOD)
        {
            AcpiGbl_NumExternalMethods++;
            if (NextExternal->Flags & ACPI_EXT_RESOLVED_REFERENCE)
            {
                AcpiGbl_ResolvedExternalMethods++;
            }
        }
```

```
1049         NextExternal = NextExternal->Next;
1050     }
1051
1052     /* Check if any control methods were unresolved */
1053
1054     AcpiDmUnresolvedWarning (1);
1055
1056     /* Emit any unresolved method externals in a single text block */
1057
1058     NextExternal = AcpiGbl_ExternalList;
1059     while (NextExternal)
1060     {
1061         if ((NextExternal->Type == ACPI_TYPE_METHOD) &&
1062             (!(NextExternal->Flags & ACPI_EXT_RESOLVED_REFERENCE)))
1063         {
1064             AcpiOsPrintf ("    External (%s%s",
1065                 NextExternal->Path,
1066                 AcpiDmGetObjectTypeName (NextExternal->Type));
1067
1068             AcpiOsPrintf (
1069                 ")    // Warning: Unresolved Method, "
1070                 "guessing %u arguments (may be incorrect, see warning above)\n",
1071                 NextExternal->Value);
1072
1073             NextExternal->Flags |= ACPI_EXT_EXTERNAL_EMITTED;
1074         }
1075
1076         NextExternal = NextExternal->Next;
1077     }
1078
1079     AcpiOsPrintf ("\n");
1080
1081
1082     /* Emit externals that were imported from a file */
1083
1084     if (Gbl_ExternalRefFilename)
1085     {
1086         AcpiOsPrintf (
1087             "    /*\n     * External declarations that were imported from\n"
1088             "     * the reference file [%s]\n     */\n",
1089             Gbl_ExternalRefFilename);
1090
1091         NextExternal = AcpiGbl_ExternalList;
1092         while (NextExternal)
1093         {
1094             if (!(NextExternal->Flags & ACPI_EXT_EXTERNAL_EMITTED) &&
1095                 (NextExternal->Flags & ACPI_EXT_ORIGIN_FROM_FILE))
1096             {
1097                 AcpiOsPrintf ("    External (%s%s",
1098                     NextExternal->Path,
1099                     AcpiDmGetObjectTypeName (NextExternal->Type));
1100
1101                 if (NextExternal->Type == ACPI_TYPE_METHOD)
1102                 {
1103                     AcpiOsPrintf (")    // %u Arguments\n",
1104                         NextExternal->Value);
1105                 }
1106                 else
1107                 {
1108                     AcpiOsPrintf (")\n");
1109                 }
1110                 NextExternal->Flags |= ACPI_EXT_EXTERNAL_EMITTED;
1111             }
1112
1113             NextExternal = NextExternal->Next;
1114         }
```



```
1115
1116         AcpiOsPrintf ("\n");
1117     }
1118
1119     /*
1120      * Walk the list of externals found during the AML parsing
1121      */
1122     while (AcpiGbl_ExternalList)
1123     {
1124         if (!(AcpiGbl_ExternalList->Flags & ACPI_EXT_EXTERNAL_EMITTED))
1125         {
1126             AcpiOsPrintf ("    External (%s%s",
1127                 AcpiGbl_ExternalList->Path,
1128                 AcpiDmGetObjectTypeName (AcpiGbl_ExternalList->Type));
1129
1130             /* For methods, add a comment with the number of arguments */
1131
1132             if (AcpiGbl_ExternalList->Type == ACPI_TYPE_METHOD)
1133             {
1134                 AcpiOsPrintf (")    // %u Arguments\n",
1135                     AcpiGbl_ExternalList->Value);
1136             }
1137             else
1138             {
1139                 AcpiOsPrintf (")\n");
1140             }
1141         }
1142
1143         /* Free this external info block and move on to next external */
1144
1145         NextExternal = AcpiGbl_ExternalList->Next;
1146         if (AcpiGbl_ExternalList->Flags & ACPI_EXT_INTERNAL_PATH_ALLOCATED)
1147         {
1148             ACPI_FREE (AcpiGbl_ExternalList->InternalPath);
1149         }
1150
1151         ACPI_FREE (AcpiGbl_ExternalList->Path);
1152         ACPI_FREE (AcpiGbl_ExternalList);
1153         AcpiGbl_ExternalList = NextExternal;
1154     }
1155
1156     AcpiOsPrintf ("\n");
1157 }
1158
1159
1160 /*******************************************************************************
1161  *
1162  * FUNCTION:    AcpiDmUnresolvedWarning
1163  *
1164  * PARAMETERS:  Type                - Where to output the warning.
1165  *                                    0 means write to stderr
1166  *                                    1 means write to AcpiOsPrintf
1167  *
1168  * RETURN:      None
1169  *
1170  * DESCRIPTION: Issue warning message if there are unresolved external control
1171  *              methods within the disassembly.
1172  *
1173  ******************************************************************************/
1174
1175 #if 0
1176 Summary of the external control method problem:
1177
1178 When the -e option is used with disassembly, the various SSDTs are simply
1179 loaded into a global namespace for the disassembler to use in order to
1180 resolve control method references (invocations).
```

```
The disassembler tracks any such references, and will emit an External()
statement for these types of methods, with the proper number of arguments .

Without the SSDTs, the AML does not contain enough information to properly
disassemble the control method invocation -- because the disassembler does
not know how many arguments to parse.

An example: Assume we have two control methods. ABCD has one argument, and
EFGH has zero arguments. Further, we have two additional control methods
that invoke ABCD and EFGH, named T1 and T2:

    Method (ABCD, 1)
    {
    }
    Method (EFGH, 0)
    {
    }
    Method (T1)
    {
        ABCD (Add (2, 7, Local0))
    }
    Method (T2)
    {
        EFGH ()
        Add (2, 7, Local0)
    }

Here is the AML code that is generated for T1 and T2:

     185:      Method (T1)

0000034C:  14 10 54 31 5F 5F 00 ...    "..T1__."

     186:      {
     187:          ABCD (Add (2, 7, Local0))

00000353:  41 42 43 44 .............    "ABCD"
00000357:  72 0A 02 0A 07 60 ......    "r....`"

     188:      }

     190:      Method (T2)

0000035D:  14 10 54 32 5F 5F 00 ...    "..T2__."

     191:      {
     192:          EFGH ()

00000364:  45 46 47 48 .............    "EFGH"

     193:          Add (2, 7, Local0)

00000368:  72 0A 02 0A 07 60 ......    "r....`"
     194:      }

Note that the AML code for T1 and T2 is essentially identical. When
disassembling this code, the methods ABCD and EFGH must be known to the
disassembler, otherwise it does not know how to handle the method invocations.

In other words, if ABCD and EFGH are actually external control methods
appearing in an SSDT, the disassembler does not know what to do unless
the owning SSDT has been loaded via the -e option.
#endif

void
```



```
AcpiDmUnresolvedWarning (
    UINT8                   Type)
{

    if (!AcpiGbl_NumExternalMethods)
    {
        return;
    }

    if (Type)
    {
        if (!AcpiGbl_ExternalFileList)
        {
            /* The -e option was not specified */

           AcpiOsPrintf ("    /*\n"
                "     * iASL Warning: There were %u external control methods fou
                "     * disassembly, but additional ACPI tables to resolve these
                "     * were not specified. This resulting disassembler output f
                "     * compile because the disassembler did not know how many a
                "     * to assign to these methods. To specify the tables needed
                "     * external control method references, use the one of the f
                "     * example iASL invocations:\n"
                "     *     iasl -e <ssdt1.aml,ssdt2.aml...> -d <dsdt.aml>\n"
                "     *     iasl -e <dsdt.aml,ssdt2.aml...> -d <ssdt1.aml>\n"
                "     */\n",
                AcpiGbl_NumExternalMethods);
        }
        else if (AcpiGbl_NumExternalMethods != AcpiGbl_ResolvedExternalMethods)
        {
            /* The -e option was specified, but there are still some unresolved

            AcpiOsPrintf ("    /*\n"
                "     * iASL Warning: There were %u external control methods fou
                "     * disassembly, but only %u %s resolved (%u unresolved). Ad
                "     * ACPI tables are required to properly disassemble the cod
                "     * resulting disassembler output file may not compile becau
                "     * disassembler did not know how many arguments to assign t
                "     * unresolved methods.\n"
                "     */\n",
                AcpiGbl_NumExternalMethods, AcpiGbl_ResolvedExternalMethods,
                (AcpiGbl_ResolvedExternalMethods > 1 ? "were" : "was"),
                (AcpiGbl_NumExternalMethods - AcpiGbl_ResolvedExternalMethods));
        }
    }
    else
    {
        if (!AcpiGbl_ExternalFileList)
        {
            /* The -e option was not specified */

            fprintf (stderr, "\n"
                "iASL Warning: There were %u external control methods found duri
                "disassembly, but additional ACPI tables to resolve these extern
                "were not specified. The resulting disassembler output file may
                "compile because the disassembler did not know how many argument
                "to assign to these methods. To specify the tables needed to res
                "external control method references, use the one of the followin
                "example iASL invocations:\n"
                "    iasl -e <ssdt1.aml,ssdt2.aml...> -d <dsdt.aml>\n"
                "    iasl -e <dsdt.aml,ssdt2.aml...> -d <ssdt1.aml>\n",
                AcpiGbl_NumExternalMethods);
        }
        else if (AcpiGbl_NumExternalMethods != AcpiGbl_ResolvedExternalMethods)
        {
            /* The -e option was specified, but there are still some unresolved
```

```
1314             fprintf (stderr, "\n"
1315                 "iASL Warning: There were %u external control methods found duri
1316                 "disassembly, but only %u %s resolved (%u unresolved). Additiona
1317                 "ACPI tables are required to properly disassemble the code. The\
1318                 "resulting disassembler output file may not compile because the\
1319                 "disassembler did not know how many arguments to assign to the\n
1320                 "unresolved methods.\n",
1321                 AcpiGbl_NumExternalMethods, AcpiGbl_ResolvedExternalMethods,
1322                 (AcpiGbl_ResolvedExternalMethods > 1 ? "were" : "was"),
1323                 (AcpiGbl_NumExternalMethods - AcpiGbl_ResolvedExternalMethods));
1324         }
1325     }
1326 }
```

```
**********************************************************
   33385 Thu Dec 26 13:48:24 2013
new/usr/src/common/acpica/common/dmrestag.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmrestag - Add tags to resource descriptors (Application-level)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acdisasm.h"
#include "acnamesp.h"
#include "amlcode.h"

/* This module used for application-level code only */

#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmrestag")

/* Local prototypes */

static void
AcpiDmUpdateResourceName (
```



```
    ACPI_NAMESPACE_NODE     *ResourceNode);

static char *
AcpiDmSearchTagList (
    UINT32                  BitIndex,
    const ACPI_RESOURCE_TAG *TagList);

static char *
AcpiDmGetResourceTag (
    UINT32                  BitIndex,
    AML_RESOURCE            *Resource,
    UINT8                   ResourceIndex);

static char *
AcpiGetTagPathname (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_NAMESPACE_NODE     *BufferNode,
    ACPI_NAMESPACE_NODE     *ResourceNode,
    UINT32                  BitIndex);

static ACPI_NAMESPACE_NODE *
AcpiDmGetResourceNode (
    ACPI_NAMESPACE_NODE     *BufferNode,
    UINT32                  BitIndex);

static ACPI_STATUS
AcpiDmAddResourceToNamespace (
    UINT8                   *Aml,
    UINT32                  Length,
    UINT32                  Offset,
    UINT8                   ResourceIndex,
    void                    **Context);

static void
AcpiDmAddResourcesToNamespace (
    ACPI_NAMESPACE_NODE     *BufferNode,
    ACPI_PARSE_OBJECT       *Op);


/******************************************************************************
 *
 * Resource Tag tables
 *
 * These are the predefined tags that refer to elements of a resource
 * descriptor. Each name and offset is defined in the ACPI specification.
 *
 * Each table entry contains the bit offset of the field and the associated
 * name.
 *
 *****************************************************************************/

static const ACPI_RESOURCE_TAG      AcpiDmIrqTags[] =
{
    {( 1 * 8),      ACPI_RESTAG_INTERRUPT},
    {( 3 * 8) + 0,  ACPI_RESTAG_INTERRUPTTYPE},
    {( 3 * 8) + 3,  ACPI_RESTAG_INTERRUPTLEVEL},
    {( 3 * 8) + 4,  ACPI_RESTAG_INTERRUPTSHARE},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmDmaTags[] =
{
    {( 1 * 8),      ACPI_RESTAG_DMA},
    {( 2 * 8) + 0,  ACPI_RESTAG_XFERTYPE},
    {( 2 * 8) + 2,  ACPI_RESTAG_BUSMASTER},
    {( 2 * 8) + 5,  ACPI_RESTAG_DMATYPE},
```

```
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmIoTags[] =
{
    {( 1 * 8) + 0,  ACPI_RESTAG_DECODE},
    {( 2 * 8),      ACPI_RESTAG_MINADDR},
    {( 4 * 8),      ACPI_RESTAG_MAXADDR},
    {( 6 * 8),      ACPI_RESTAG_ALIGNMENT},
    {( 7 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmFixedIoTags[] =
{
    {( 1 * 8),      ACPI_RESTAG_BASEADDRESS},
    {( 3 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmFixedDmaTags[] =
{
    {( 1 * 8),      ACPI_RESTAG_DMA},
    {( 3 * 8),      ACPI_RESTAG_DMATYPE},
    {( 5 * 8),      ACPI_RESTAG_XFERTYPE},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmMemory24Tags[] =
{
    {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYPE},
    {( 4 * 8),      ACPI_RESTAG_MINADDR},
    {( 6 * 8),      ACPI_RESTAG_MAXADDR},
    {( 8 * 8),      ACPI_RESTAG_ALIGNMENT},
    {(10 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmRegisterTags[] =
{
    {( 3 * 8),      ACPI_RESTAG_ADDRESSSPACE},
    {( 4 * 8),      ACPI_RESTAG_REGISTERBITWIDTH},
    {( 5 * 8),      ACPI_RESTAG_REGISTERBITOFFSET},
    {( 6 * 8),      ACPI_RESTAG_ACCESSSIZE},
    {( 7 * 8),      ACPI_RESTAG_ADDRESS},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmMemory32Tags[] =
{
    {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYPE},
    {( 4 * 8),      ACPI_RESTAG_MINADDR},
    {( 8 * 8),      ACPI_RESTAG_MAXADDR},
    {(12 * 8),      ACPI_RESTAG_ALIGNMENT},
    {(16 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmFixedMemory32Tags[] =
{
    {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYPE},
    {( 4 * 8),      ACPI_RESTAG_BASEADDRESS},
    {( 8 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};
```



```
static const ACPI_RESOURCE_TAG      AcpiDmInterruptTags[] =
{
    {( 3 * 8) + 1,  ACPI_RESTAG_INTERRUPTTYPE},
    {( 3 * 8) + 2,  ACPI_RESTAG_INTERRUPTLEVEL},
    {( 3 * 8) + 3,  ACPI_RESTAG_INTERRUPTSHARE},
    {( 5 * 8),      ACPI_RESTAG_INTERRUPT},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmAddress16Tags[] =
{
    {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
    {( 4 * 8) + 2,  ACPI_RESTAG_MINTYPE},
    {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYPE},
    {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
    {( 8 * 8),      ACPI_RESTAG_MINADDR},
    {(10 * 8),      ACPI_RESTAG_MAXADDR},
    {(12 * 8),      ACPI_RESTAG_TRANSLATION},
    {(14 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmAddress32Tags[] =
{
    {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
    {( 4 * 8) + 2,  ACPI_RESTAG_MINTYPE},
    {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYPE},
    {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
    {(10 * 8),      ACPI_RESTAG_MINADDR},
    {(14 * 8),      ACPI_RESTAG_MAXADDR},
    {(18 * 8),      ACPI_RESTAG_TRANSLATION},
    {(22 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmAddress64Tags[] =
{
    {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
    {( 4 * 8) + 2,  ACPI_RESTAG_MINTYPE},
    {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYPE},
    {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
    {(14 * 8),      ACPI_RESTAG_MINADDR},
    {(22 * 8),      ACPI_RESTAG_MAXADDR},
    {(30 * 8),      ACPI_RESTAG_TRANSLATION},
    {(38 * 8),      ACPI_RESTAG_LENGTH},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmExtendedAddressTags[] =
{
    {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
    {( 4 * 8) + 2,  ACPI_RESTAG_MINTYPE},
    {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYPE},
    {( 8 * 8),      ACPI_RESTAG_GRANULARITY},
    {(16 * 8),      ACPI_RESTAG_MINADDR},
    {(24 * 8),      ACPI_RESTAG_MAXADDR},
    {(32 * 8),      ACPI_RESTAG_TRANSLATION},
    {(40 * 8),      ACPI_RESTAG_LENGTH},
    {(48 * 8),      ACPI_RESTAG_TYPESPECIFICATTRIBUTES},
    {0,             NULL}
};

/* Subtype tables for GPIO descriptors */

static const ACPI_RESOURCE_TAG      AcpiDmGpioIntTags[] =
{
```

```
    {( 7 * 8) + 0,  ACPI_RESTAG_MODE},
    {( 7 * 8) + 1,  ACPI_RESTAG_POLARITY},
    {( 7 * 8) + 3,  ACPI_RESTAG_INTERRUPTSHARE},
    {( 9 * 8),      ACPI_RESTAG_PINCONFIG},
    {(10 * 8),      ACPI_RESTAG_DRIVESTRENGTH},
    {(12 * 8),      ACPI_RESTAG_DEBOUNCETIME},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmGpioIoTags[] =
{
    {( 7 * 8) + 0,  ACPI_RESTAG_IORESTRICTION},
    {( 7 * 8) + 3,  ACPI_RESTAG_INTERRUPTSHARE},
    {( 9 * 8),      ACPI_RESTAG_PINCONFIG},
    {(10 * 8),      ACPI_RESTAG_DRIVESTRENGTH},
    {(12 * 8),      ACPI_RESTAG_DEBOUNCETIME},
    {0,             NULL}
};

/* Subtype tables for SerialBus descriptors */

static const ACPI_RESOURCE_TAG      AcpiDmI2cSerialBusTags[] =
{
    {( 6 * 8) + 0,  ACPI_RESTAG_SLAVEMODE},
    {( 7 * 8) + 0,  ACPI_RESTAG_MODE},
    {(12 * 8),      ACPI_RESTAG_SPEED},
    {(16 * 8),      ACPI_RESTAG_ADDRESS},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmSpiSerialBusTags[] =
{
    {( 6 * 8) + 0,  ACPI_RESTAG_SLAVEMODE},
    {( 7 * 8) + 0,  ACPI_RESTAG_MODE},
    {( 7 * 8) + 1,  ACPI_RESTAG_DEVICEPOLARITY},
    {(12 * 8),      ACPI_RESTAG_SPEED},
    {(16 * 8),      ACPI_RESTAG_LENGTH},
    {(17 * 8),      ACPI_RESTAG_PHASE},
    {(18 * 8),      ACPI_RESTAG_POLARITY},
    {(19 * 8),      ACPI_RESTAG_ADDRESS},
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmUartSerialBusTags[] =
{
    {( 6 * 8) + 0,  ACPI_RESTAG_SLAVEMODE}, /* Note: not part of original macro
    {( 7 * 8) + 0,  ACPI_RESTAG_FLOWCONTROL},
    {( 7 * 8) + 2,  ACPI_RESTAG_STOPBITS},
    {( 7 * 8) + 4,  ACPI_RESTAG_LENGTH},
    {( 7 * 8) + 7,  ACPI_RESTAG_ENDIANNESS},
    {(12 * 8),      ACPI_RESTAG_SPEED},
    {(16 * 8),      ACPI_RESTAG_LENGTH_RX},
    {(18 * 8),      ACPI_RESTAG_LENGTH_TX},
    {(20 * 8),      ACPI_RESTAG_PARITY},
    {(21 * 8),      ACPI_RESTAG_LINE},
    {0,             NULL}
};

/* Subtype tables for Address descriptor type-specific flags */

static const ACPI_RESOURCE_TAG      AcpiDmMemoryFlagTags[] =
{
    {( 5 * 8) + 0,  ACPI_RESTAG_READWRITETYPE},
    {( 5 * 8) + 1,  ACPI_RESTAG_MEMTYPE},
    {( 5 * 8) + 3,  ACPI_RESTAG_MEMATTRIBUTES},
    {( 5 * 8) + 5,  ACPI_RESTAG_TYPE},
```



```
    {0,             NULL}
};

static const ACPI_RESOURCE_TAG      AcpiDmIoFlagTags[] =
{
    {( 5 * 8) + 0,  ACPI_RESTAG_RANGETYPE},
    {( 5 * 8) + 4,  ACPI_RESTAG_TYPE},
    {( 5 * 8) + 5,  ACPI_RESTAG_TRANSTYPE},
    {0,             NULL}
};


/*
 * Dispatch table used to obtain the correct tag table for a descriptor.
 *
 * A NULL in this table means one of three things:
 * 1) The descriptor ID is reserved and invalid
 * 2) The descriptor has no tags associated with it
 * 3) The descriptor has subtypes and a separate table will be used.
 */
static const ACPI_RESOURCE_TAG      *AcpiGbl_ResourceTags[] =
{
    /* Small descriptors */

    NULL,                           /* 0x00, Reserved */
    NULL,                           /* 0x01, Reserved */
    NULL,                           /* 0x02, Reserved */
    NULL,                           /* 0x03, Reserved */
    AcpiDmIrqTags,                  /* 0x04, ACPI_RESOURCE_NAME_IRQ_FORMAT */
    AcpiDmDmaTags,                  /* 0x05, ACPI_RESOURCE_NAME_DMA_FORMAT */
    NULL,                           /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT
    NULL,                           /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
    AcpiDmIoTags,                   /* 0x08, ACPI_RESOURCE_NAME_IO_PORT */
    AcpiDmFixedIoTags,              /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO_PORT */
    AcpiDmFixedDmaTags,             /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
    NULL,                           /* 0x0B, Reserved */
    NULL,                           /* 0x0C, Reserved */
    NULL,                           /* 0x0D, Reserved */
    NULL,                           /* 0x0E, ACPI_RESOURCE_NAME_SMALL_VENDOR */
    NULL,                           /* 0x0F, ACPI_RESOURCE_NAME_END_TAG (not use

    /* Large descriptors */

    NULL,                           /* 0x00, Reserved */
    AcpiDmMemory24Tags,             /* 0x01, ACPI_RESOURCE_NAME_MEMORY_24 */
    AcpiDmRegisterTags,             /* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER
    NULL,                           /* 0x03, Reserved */
    NULL,                           /* 0x04, ACPI_RESOURCE_NAME_LARGE_VENDOR */
    AcpiDmMemory32Tags,             /* 0x05, ACPI_RESOURCE_NAME_MEMORY_32 */
    AcpiDmFixedMemory32Tags,        /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY_32
    AcpiDmAddress32Tags,            /* 0x07, ACPI_RESOURCE_NAME_DWORD_ADDRESS_SP
    AcpiDmAddress16Tags,            /* 0x08, ACPI_RESOURCE_NAME_WORD_ADDRESS_SPA
    AcpiDmInterruptTags,            /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_XRUPT *
    AcpiDmAddress64Tags,            /* 0x0A, ACPI_RESOURCE_NAME_QWORD_ADDRESS_SP
    AcpiDmExtendedAddressTags,      /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
    NULL,                           /* 0x0C, ACPI_RESOURCE_NAME_GPIO - Use Subty
    NULL,                           /* 0x0D, Reserved */
    NULL                            /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS - Use
};

/* GPIO Subtypes */

static const ACPI_RESOURCE_TAG      *AcpiGbl_GpioResourceTags[] =
{
    AcpiDmGpioIntTags,              /* 0x00 Interrupt Connection */
    AcpiDmGpioIoTags                /* 0x01 I/O Connection */
```

```
391 };

393 /* Serial Bus Subtypes */

395 static const ACPI_RESOURCE_TAG      *AcpiGbl_SerialResourceTags[] =
396 {
397     NULL,                           /* 0x00 Reserved */
398     AcpiDmI2cSerialBusTags,         /* 0x01 I2C SerialBus */
399     AcpiDmSpiSerialBusTags,         /* 0x02 SPI SerialBus */
400     AcpiDmUartSerialBusTags         /* 0x03 UART SerialBus */
401 };

403 /*
404  * Globals used to generate unique resource descriptor names. We use names that
405  * start with underscore and a prefix letter that is not used by other ACPI
406  * reserved names. To this, we append hex 0x00 through 0xFF. These 5 prefixes
407  * allow for 5*256 = 1280 unique names, probably sufficient for any single ASL
408  * file. If this becomes too small, we can use alpha+numerals for a total
409  * of 5*36*36 = 6480.
410  */
411 #define ACPI_NUM_RES_PREFIX     5

413 static UINT32                   AcpiGbl_NextResourceId = 0;
414 static UINT8                    AcpiGbl_NextPrefix = 0;
415 static char                     AcpiGbl_Prefix[ACPI_NUM_RES_PREFIX] =
416                                     {'Y','Z','J','K','X'};


419 /*****************************************************************************
420  *
421  * FUNCTION:    AcpiDmCheckResourceReference
422  *
423  * PARAMETERS:  Op                  - Parse Op for the AML opcode
424  *              WalkState           - Current walk state (with valid scope)
425  *
426  * RETURN:      None
427  *
428  * DESCRIPTION: Convert a reference to a resource descriptor to a symbolic
429  *              reference if possible
430  *
431  * NOTE:        Bit index is used to transparently handle both resource bit
432  *              fields and byte fields.
433  *
434  *****************************************************************************/

436 void
437 AcpiDmCheckResourceReference (
438     ACPI_PARSE_OBJECT       *Op,
439     ACPI_WALK_STATE         *WalkState)
440 {
441     ACPI_STATUS             Status;
442     ACPI_PARSE_OBJECT       *BufferNameOp;
443     ACPI_PARSE_OBJECT       *IndexOp;
444     ACPI_NAMESPACE_NODE     *BufferNode;
445     ACPI_NAMESPACE_NODE     *ResourceNode;
446     const ACPI_OPCODE_INFO  *OpInfo;
447     UINT32                  BitIndex;


450     /* We are only interested in the CreateXxxxField opcodes */

452     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
453     if (OpInfo->Type != AML_TYPE_CREATE_FIELD)
454     {
455         return;
456     }
```



```
458     /* Get the buffer term operand */

460     BufferNameOp = AcpiPsGetDepthNext (NULL, Op);

462     /* Must be a named buffer, not an arg or local or method call */

464     if (BufferNameOp->Common.AmlOpcode != AML_INT_NAMEPATH_OP)
465     {
466         return;
467     }

469     /* Get the Index term, must be an integer constant to convert */

471     IndexOp = BufferNameOp->Common.Next;

473     /* Major cheat: The Node field is also used for the Tag ptr. Clear it now */

475     IndexOp->Common.Node = NULL;

477     OpInfo = AcpiPsGetOpcodeInfo (IndexOp->Common.AmlOpcode);
478     if (OpInfo->ObjectType != ACPI_TYPE_INTEGER)
479     {
480         return;
481     }

483     /* Get the bit offset of the descriptor within the buffer */

485     if ((Op->Common.AmlOpcode == AML_CREATE_BIT_FIELD_OP) ||
486         (Op->Common.AmlOpcode == AML_CREATE_FIELD_OP))
487     {
488         /* Index operand is a bit offset */

490         BitIndex = (UINT32) IndexOp->Common.Value.Integer;
491     }
492     else
493     {
494         /* Index operand is a byte offset, convert to bits */

496         BitIndex = (UINT32) ACPI_MUL_8 (IndexOp->Common.Value.Integer);
497     }

499     /* Lookup the buffer in the namespace */

501     Status = AcpiNsLookup (WalkState->ScopeInfo,
502                 BufferNameOp->Common.Value.String, ACPI_TYPE_BUFFER,
503                 ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT, WalkState,
504                 &BufferNode);
505     if (ACPI_FAILURE (Status))
506     {
507         return;
508     }

510     /* Validate object type, we must have a buffer */

512     if (BufferNode->Type != ACPI_TYPE_BUFFER)
513     {
514         return;
515     }

517     /* Find the resource descriptor node corresponding to the index */

519     ResourceNode = AcpiDmGetResourceNode (BufferNode, BitIndex);
520     if (!ResourceNode)
521     {
522         return;
```

```
    }

    /* Translate the Index to a resource tag pathname */

    AcpiGetTagPathname (IndexOp, BufferNode, ResourceNode, BitIndex);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmGetResourceNode
 *
 * PARAMETERS:  BufferNode          - Node for the parent buffer
 *              BitIndex            - Index into the resource descriptor
 *
 * RETURN:      Namespace node for the resource descriptor. NULL if not found
 *
 * DESCRIPTION: Find a resource descriptor that corresponds to the bit index
 *
 ******************************************************************************/

static ACPI_NAMESPACE_NODE *
AcpiDmGetResourceNode (
    ACPI_NAMESPACE_NODE     *BufferNode,
    UINT32                  BitIndex)
{
    ACPI_NAMESPACE_NODE     *Node;
    UINT32                  ByteIndex = ACPI_DIV_8 (BitIndex);


    /*
     * Child list contains an entry for each resource descriptor. Find
     * the descriptor that corresponds to the Index.
     *
     * If there are no children, this is not a resource template
     */
    Node = BufferNode->Child;
    while (Node)
    {
        /*
         * Check if the Index falls within this resource.
         *
         * Value contains the resource offset, Object contains the resource
         * length (both in bytes)
         */
        if ((ByteIndex >= Node->Value) &&
            (ByteIndex < (Node->Value + Node->Length)))
        {
            return (Node);
        }

        Node = Node->Peer;
    }

    return (NULL);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiGetTagPathname
 *
 * PARAMETERS:  BufferNode          - Node for the parent buffer
 *              ResourceNode        - Node for a resource descriptor
 *              BitIndex            - Index into the resource descriptor
 *
```



```
 * RETURN:      Full pathname for a resource tag. NULL if no match.
 *              Path is returned in AML (packed) format.
 *
 * DESCRIPTION: Convert a BitIndex into a symbolic resource tag (full pathname)
 *
 ******************************************************************************/

static char *
AcpiGetTagPathname (
    ACPI_PARSE_OBJECT       *IndexOp,
    ACPI_NAMESPACE_NODE     *BufferNode,
    ACPI_NAMESPACE_NODE     *ResourceNode,
    UINT32                  BitIndex)
{
    ACPI_STATUS             Status;
    UINT32                  ResourceBitIndex;
    UINT8                   ResourceTableIndex;
    ACPI_SIZE               RequiredSize;
    char                    *Pathname;
    AML_RESOURCE            *Aml;
    ACPI_PARSE_OBJECT       *Op;
    char                    *InternalPath;
    char                    *Tag;


    /* Get the Op that contains the actual buffer data */

    Op = BufferNode->Op->Common.Value.Arg;
    Op = Op->Common.Next;
    if (!Op)
    {
        return (NULL);
    }

    /* Get the individual resource descriptor and validate it */

    Aml = ACPI_CAST_PTR (AML_RESOURCE,
            &Op->Named.Data[ResourceNode->Value]);

    Status = AcpiUtValidateResource (NULL, Aml, &ResourceTableIndex);
    if (ACPI_FAILURE (Status))
    {
        return (NULL);
    }

    /* Get offset into this descriptor (from offset into entire buffer) */

    ResourceBitIndex = BitIndex - ACPI_MUL_8 (ResourceNode->Value);

    /* Get the tag associated with this resource descriptor and offset */

    Tag = AcpiDmGetResourceTag (ResourceBitIndex, Aml, ResourceTableIndex);
    if (!Tag)
    {
        return (NULL);
    }

    /*
     * Now that we know that we have a reference that can be converted to a
     * symbol, change the name of the resource to a unique name.
     */
    AcpiDmUpdateResourceName (ResourceNode);

    /* Get the full pathname to the parent buffer */

    RequiredSize = AcpiNsGetPathnameLength (BufferNode);
```

```
655     if (!RequiredSize)
656     {
657         return (NULL);
658     }

660     Pathname = ACPI_ALLOCATE_ZEROED (RequiredSize + ACPI_PATH_SEGMENT_LENGTH);
661     if (!Pathname)
662     {
663         return (NULL);
664     }

666     Status = AcpiNsBuildExternalPath (BufferNode, RequiredSize, Pathname);
667     if (ACPI_FAILURE (Status))
668     {
669         ACPI_FREE (Pathname);
670         return (NULL);
671     }

673     /*
674      * Create the full path to the resource and tag by: remove the buffer name,
675      * append the resource descriptor name, append a dot, append the tag name.
676      *
677      * TBD: Always using the full path is a bit brute force, the path can be
678      * often be optimized with carats (if the original buffer namepath is a
679      * single nameseg). This doesn't really matter, because these paths do not
680      * end up in the final compiled AML, it's just an appearance issue for the
681      * disassembled code.
682      */
683     Pathname[ACPI_STRLEN (Pathname) - ACPI_NAME_SIZE] = 0;
684     ACPI_STRNCAT (Pathname, ResourceNode->Name.Ascii, ACPI_NAME_SIZE);
685     ACPI_STRCAT (Pathname, ".");
686     ACPI_STRNCAT (Pathname, Tag, ACPI_NAME_SIZE);

688     /* Internalize the namepath to AML format */

690     AcpiNsInternalizeName (Pathname, &InternalPath);
691     ACPI_FREE (Pathname);

693     /* Update the Op with the symbol */

695     AcpiPsInitOp (IndexOp, AML_INT_NAMEPATH_OP);
696     IndexOp->Common.Value.String = InternalPath;

698     /* We will need the tag later. Cheat by putting it in the Node field */

700     IndexOp->Common.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Tag);
701     return (InternalPath);
702 }


705 /*******************************************************************************
706  *
707  * FUNCTION:    AcpiDmUpdateResourceName
708  *
709  * PARAMETERS:  ResourceNode        - Node for a resource descriptor
710  *
711  * RETURN:      Stores new name in the ResourceNode
712  *
713  * DESCRIPTION: Create a new, unique name for a resource descriptor. Used by
714  *              both the disassembly of the descriptor itself and any symbolic
715  *              references to the descriptor. Ignored if a unique name has
716  *              already been assigned to the resource.
717  *
718  * NOTE: Single threaded, suitable for applications only!
719  *
720  ******************************************************************************/
```



```
722 static void
723 AcpiDmUpdateResourceName (
724     ACPI_NAMESPACE_NODE     *ResourceNode)
725 {
726     char                    Name[ACPI_NAME_SIZE];


729     /* Ignore if a unique name has already been assigned */

731     if (ResourceNode->Name.Integer != ACPI_DEFAULT_RESNAME)
732     {
733         return;
734     }

736     /* Generate a new ACPI name for the descriptor */

738     Name[0] = '_';
739     Name[1] = AcpiGbl_Prefix[AcpiGbl_NextPrefix];
740     Name[2] = AcpiUtHexToAsciiChar ((UINT64) AcpiGbl_NextResourceId, 4);
741     Name[3] = AcpiUtHexToAsciiChar ((UINT64) AcpiGbl_NextResourceId, 0);

743     /* Update globals for next name */

745     AcpiGbl_NextResourceId++;
746     if (AcpiGbl_NextResourceId >= 256)
747     {
748         AcpiGbl_NextResourceId = 0;
749         AcpiGbl_NextPrefix++;
750         if (AcpiGbl_NextPrefix > ACPI_NUM_RES_PREFIX)
751         {
752             AcpiGbl_NextPrefix = 0;
753         }
754     }

756     /* Change the resource descriptor name */

758     ResourceNode->Name.Integer = *ACPI_CAST_PTR (UINT32, &Name[0]);
759 }


762 /*******************************************************************************
763  *
764  * FUNCTION:    AcpiDmGetResourceTag
765  *
766  * PARAMETERS:  BitIndex            - Index into the resource descriptor
767  *              Resource            - Pointer to the raw resource data
768  *              ResourceIndex       - Index correspoinding to the resource type
769  *
770  * RETURN:      Pointer to the resource tag (ACPI_NAME). NULL if no match.
771  *
772  * DESCRIPTION: Convert a BitIndex into a symbolic resource tag.
773  *
774  * Note: ResourceIndex should be previously validated and guaranteed to ve
775  *       valid.
776  *
777  ******************************************************************************/

779 static char *
780 AcpiDmGetResourceTag (
781     UINT32                  BitIndex,
782     AML_RESOURCE            *Resource,
783     UINT8                   ResourceIndex)
784 {
785     const ACPI_RESOURCE_TAG *TagList;
786     char                    *Tag = NULL;
```

```
789     /* Get the tag list for this resource descriptor type */

791     TagList = AcpiGbl_ResourceTags[ResourceIndex];

793     /*
794      * Handle descriptors that have multiple subtypes
795      */
796     switch (Resource->DescriptorType)
797     {
798     case ACPI_RESOURCE_NAME_ADDRESS16:
799     case ACPI_RESOURCE_NAME_ADDRESS32:
800     case ACPI_RESOURCE_NAME_ADDRESS64:
801     case ACPI_RESOURCE_NAME_EXTENDED_ADDRESS64:
802         /*
803          * Subtype differentiation is the flags.
804          * Kindof brute force, but just blindly search for an index match
805          */
806         if (Resource->Address.ResourceType == ACPI_ADDRESS_TYPE_MEMORY_RANGE)
807         {
808             Tag = AcpiDmSearchTagList (BitIndex, AcpiDmMemoryFlagTags);
809         }
810         else if (Resource->Address.ResourceType == ACPI_ADDRESS_TYPE_IO_RANGE)
811         {
812             Tag = AcpiDmSearchTagList (BitIndex, AcpiDmIoFlagTags);
813         }

815         /* If we found a match, all done. Else, drop to normal search below */

817         if (Tag)
818         {
819             return (Tag);
820         }
821         break;

823     case ACPI_RESOURCE_NAME_GPIO:

825         /* GPIO connection has 2 subtypes: Interrupt and I/O */

827         if (Resource->Gpio.ConnectionType > AML_RESOURCE_MAX_GPIOTYPE)
828         {
829             return (NULL);
830         }

832         TagList = AcpiGbl_GpioResourceTags[Resource->Gpio.ConnectionType];
833         break;

835     case ACPI_RESOURCE_NAME_SERIAL_BUS:

837         /* SerialBus has 3 subtypes: I2C, SPI, and UART */

839         if ((Resource->CommonSerialBus.Type == 0) ||
840             (Resource->CommonSerialBus.Type > AML_RESOURCE_MAX_SERIALBUSTYPE))
841         {
842             return (NULL);
843         }

845         TagList = AcpiGbl_SerialResourceTags[Resource->CommonSerialBus.Type];
846         break;

848     default:

850         break;
851     }
```



```
853     /* Search for a match against the BitIndex */

855     if (TagList)
856     {
857         Tag = AcpiDmSearchTagList (BitIndex, TagList);
858     }

860     return (Tag);
861 }


864 /*******************************************************************************
865  *
866  * FUNCTION:    AcpiDmSearchTagList
867  *
868  * PARAMETERS:  BitIndex            - Index into the resource descriptor
869  *              TagList             - List to search
870  *
871  * RETURN:      Pointer to a tag (ACPI_NAME). NULL if no match found.
872  *
873  * DESCRIPTION: Search a tag list for a match to the input BitIndex. Matches
874  *              a fixed offset to a symbolic resource tag name.
875  *
876  ******************************************************************************/

878 static char *
879 AcpiDmSearchTagList (
880     UINT32                  BitIndex,
881     const ACPI_RESOURCE_TAG *TagList)
882 {

884     /*
885      * Walk the null-terminated tag list to find a matching bit offset.
886      * We are looking for an exact match.
887      */
888     for ( ; TagList->Tag; TagList++)
889     {
890         if (BitIndex == TagList->BitIndex)
891         {
892             return (TagList->Tag);
893         }
894     }

896     /* A matching offset was not found */

898     return (NULL);
899 }


902 /*******************************************************************************
903  *
904  * FUNCTION:    AcpiDmFindResources
905  *
906  * PARAMETERS:  Root                - Root of the parse tree
907  *
908  * RETURN:      None
909  *
910  * DESCRIPTION: Add all ResourceTemplate declarations to the namespace. Each
911  *              resource descriptor in each template is given a node -- used
912  *              for later conversion of resource references to symbolic refs.
913  *
914  ******************************************************************************/

916 void
917 AcpiDmFindResources (
918     ACPI_PARSE_OBJECT       *Root)
```

```
{
    ACPI_PARSE_OBJECT       *Op = Root;
    ACPI_PARSE_OBJECT       *Parent;


    /* Walk the entire parse tree */

    while (Op)
    {
        /* We are interested in Buffer() declarations */

        if (Op->Common.AmlOpcode == AML_BUFFER_OP)
        {
            /* And only declarations of the form Name (XXXX, Buffer()... ) */

            Parent = Op->Common.Parent;
            if (Parent->Common.AmlOpcode == AML_NAME_OP)
            {
                /*
                 * If the buffer is a resource template, add the individual
                 * resource descriptors to the namespace, as children of the
                 * buffer node.
                 */
                if (ACPI_SUCCESS (AcpiDmIsResourceTemplate (NULL, Op)))
                {
                    Op->Common.DisasmOpcode = ACPI_DASM_RESOURCE;
                    AcpiDmAddResourcesToNamespace (Parent->Common.Node, Op);
                }
            }
        }

        Op = AcpiPsGetDepthNext (Root, Op);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmAddResourcesToNamespace
 *
 * PARAMETERS:  BufferNode          - Node for the parent buffer
 *              Op                  - Parse op for the buffer
 *
 * RETURN:      None
 *
 * DESCRIPTION: Add an entire resource template to the namespace. Each
 *              resource descriptor is added as a namespace node.
 *
 ******************************************************************************/

static void
AcpiDmAddResourcesToNamespace (
    ACPI_NAMESPACE_NODE     *BufferNode,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *NextOp;


    /* Get to the ByteData list */

    NextOp = Op->Common.Value.Arg;
    NextOp = NextOp->Common.Next;
    if (!NextOp)
    {
        return;
    }
```



```
    /* Set Node and Op to point to each other */

    BufferNode->Op = Op;
    Op->Common.Node = BufferNode;

    /*
     * Insert each resource into the namespace
     * NextOp contains the Aml pointer and the Aml length
     */
    AcpiUtWalkAmlResources (NULL, (UINT8 *) NextOp->Named.Data,
        (ACPI_SIZE) NextOp->Common.Value.Integer,
        AcpiDmAddResourceToNamespace, (void **) BufferNode);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmAddResourceToNamespace
 *
 * PARAMETERS:  ACPI_WALK_AML_CALLBACK
 *              BufferNode              - Node for the parent buffer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Add one resource descriptor to the namespace as a child of the
 *              parent buffer. The same name is used for each descriptor. This
 *              is changed later to a unique name if the resource is actually
 *              referenced by an AML operator.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiDmAddResourceToNamespace (
    UINT8                   *Aml,
    UINT32                  Length,
    UINT32                  Offset,
    UINT8                   ResourceIndex,
    void                    **Context)
{
    ACPI_STATUS             Status;
    ACPI_GENERIC_STATE      ScopeInfo;
    ACPI_NAMESPACE_NODE     *Node;


    /* TBD: Don't need to add descriptors that have no tags defined? */

    /* Add the resource to the namespace, as child of the buffer */

    ScopeInfo.Scope.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Context);
    Status = AcpiNsLookup (&ScopeInfo, "_TMP", ACPI_TYPE_LOCAL_RESOURCE,
                ACPI_IMODE_LOAD_PASS2,
                ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE | ACPI_NS_PREFIX_I
                NULL, &Node);
    if (ACPI_FAILURE (Status))
    {
        return (AE_OK);
    }

    /* Set the name to the default, changed later if resource is referenced */

    Node->Name.Integer = ACPI_DEFAULT_RESNAME;

    /* Save the offset of the descriptor (within the original buffer) */

    Node->Value = Offset;
```

```
1051     Node->Length = Length;
1052     return (AE_OK);
1053 }
```

```
**********************************************************
   39952 Thu Dec 26 13:48:24 2013
new/usr/src/common/acpica/common/dmtable.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: dmtable - Support for ACPI tables that contain no AML code
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"
#include "actables.h"
#include "aslcompiler.h"
#include "dtcompiler.h"

/* This module used for application-level code only */

#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmtable")

/* Local Prototypes */

static void
AcpiDmCheckAscii (
```



```
    UINT8                   *Target,
    char                    *RepairedName,
    UINT32                  Count);


/* Common format strings for commented values */

#define UINT8_FORMAT        "%2.2X [%s]\n"
#define UINT16_FORMAT       "%4.4X [%s]\n"
#define UINT32_FORMAT       "%8.8X [%s]\n"
#define STRING_FORMAT       "[%s]\n"

/* These tables map a subtable type to a description string */

static const char           *AcpiDmAsfSubnames[] =
{
    "ASF Information",
    "ASF Alerts",
    "ASF Remote Control",
    "ASF RMCP Boot Options",
    "ASF Address",
    "Unknown SubTable Type"         /* Reserved */
};

static const char           *AcpiDmDmarSubnames[] =
{
    "Hardware Unit Definition",
    "Reserved Memory Region",
    "Root Port ATS Capability",
    "Remapping Hardware Static Affinity",
    "Unknown SubTable Type"         /* Reserved */
};

static const char           *AcpiDmEinjActions[] =
{
    "Begin Operation",
    "Get Trigger Table",
    "Set Error Type",
    "Get Error Type",
    "End Operation",
    "Execute Operation",
    "Check Busy Status",
    "Get Command Status",
    "Set Error Type With Address",
    "Unknown Action"
};

static const char           *AcpiDmEinjInstructions[] =
{
    "Read Register",
    "Read Register Value",
    "Write Register",
    "Write Register Value",
    "Noop",
    "Flush Cacheline",
    "Unknown Instruction"
};

static const char           *AcpiDmErstActions[] =
{
    "Begin Write Operation",
    "Begin Read Operation",
    "Begin Clear Operation",
    "End Operation",
    "Set Record Offset",
    "Execute Operation",
```

```
126     "Check Busy Status",
127     "Get Command Status",
128     "Get Record Identifier",
129     "Set Record Identifier",
130     "Get Record Count",
131     "Begin Dummy Write",
132     "Unused/Unknown Action",
133     "Get Error Address Range",
134     "Get Error Address Length",
135     "Get Error Attributes",
136     "Unknown Action"
137 };

139 static const char           *AcpiDmErstInstructions[] =
140 {
141     "Read Register",
142     "Read Register Value",
143     "Write Register",
144     "Write Register Value",
145     "Noop",
146     "Load Var1",
147     "Load Var2",
148     "Store Var1",
149     "Add",
150     "Subtract",
151     "Add Value",
152     "Subtract Value",
153     "Stall",
154     "Stall While True",
155     "Skip Next If True",
156     "GoTo",
157     "Set Source Address",
158     "Set Destination Address",
159     "Move Data",
160     "Unknown Instruction"
161 };

163 static const char           *AcpiDmHestSubnames[] =
164 {
165     "IA-32 Machine Check Exception",
166     "IA-32 Corrected Machine Check",
167     "IA-32 Non-Maskable Interrupt",
168     "Unknown SubTable Type",        /* 3 - Reserved */
169     "Unknown SubTable Type",        /* 4 - Reserved */
170     "Unknown SubTable Type",        /* 5 - Reserved */
171     "PCI Express Root Port AER",
172     "PCI Express AER (AER Endpoint)",
173     "PCI Express/PCI-X Bridge AER",
174     "Generic Hardware Error Source",
175     "Unknown SubTable Type"         /* Reserved */
176 };

178 static const char           *AcpiDmHestNotifySubnames[] =
179 {
180     "Polled",
181     "External Interrupt",
182     "Local Interrupt",
183     "SCI",
184     "NMI",
185     "CMCI",                         /* ACPI 5.0 */
186     "MCE",                          /* ACPI 5.0 */
187     "Unknown Notify Type"           /* Reserved */
188 };

190 static const char           *AcpiDmMadtSubnames[] =
191 {
```



```
192     "Processor Local APIC",         /* ACPI_MADT_TYPE_LOCAL_APIC */
193     "I/O APIC",                     /* ACPI_MADT_TYPE_IO_APIC */
194     "Interrupt Source Override",    /* ACPI_MADT_TYPE_INTERRUPT_OVERRIDE */
195     "NMI Source",                   /* ACPI_MADT_TYPE_NMI_SOURCE */
196     "Local APIC NMI",               /* ACPI_MADT_TYPE_LOCAL_APIC_NMI */
197     "Local APIC Address Override",  /* ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE */
198     "I/O SAPIC",                    /* ACPI_MADT_TYPE_IO_SAPIC */
199     "Local SAPIC",                  /* ACPI_MADT_TYPE_LOCAL_SAPIC */
200     "Platform Interrupt Sources",   /* ACPI_MADT_TYPE_INTERRUPT_SOURCE */
201     "Processor Local x2APIC",       /* ACPI_MADT_TYPE_LOCAL_X2APIC */
202     "Local x2APIC NMI",             /* ACPI_MADT_TYPE_LOCAL_X2APIC_NMI */
203     "Generic Interrupt Controller", /* ACPI_MADT_GENERIC_INTERRUPT */
204     "Generic Interrupt Distributor",/* ACPI_MADT_GENERIC_DISTRIBUTOR */
205     "Unknown SubTable Type"         /* Reserved */
206 };

208 static const char           *AcpiDmPcctSubnames[] =
209 {
210     "Generic Communications Subspace",  /* ACPI_PCCT_TYPE_GENERIC_SUBSPACE */
211     "Unknown SubTable Type"             /* Reserved */
212 };

214 static const char           *AcpiDmPmttSubnames[] =
215 {
216     "Socket",                       /* ACPI_PMTT_TYPE_SOCKET */
217     "Memory Controller",            /* ACPI_PMTT_TYPE_CONTROLLER */
218     "Physical Component (DIMM)",    /* ACPI_PMTT_TYPE_DIMM  */
219     "Unknown SubTable Type"         /* Reserved */
220 };

222 static const char           *AcpiDmSlicSubnames[] =
223 {
224     "Public Key Structure",
225     "Windows Marker Structure",
226     "Unknown SubTable Type"         /* Reserved */
227 };

229 static const char           *AcpiDmSratSubnames[] =
230 {
231     "Processor Local APIC/SAPIC Affinity",
232     "Memory Affinity",
233     "Processor Local x2APIC Affinity",
234     "Unknown SubTable Type"         /* Reserved */
235 };

237 static const char           *AcpiDmIvrsSubnames[] =
238 {
239     "Hardware Definition Block",
240     "Memory Definition Block",
241     "Unknown SubTable Type"         /* Reserved */
242 };


245 #define ACPI_FADT_PM_RESERVED       9

247 static const char           *AcpiDmFadtProfiles[] =
248 {
249     "Unspecified",
250     "Desktop",
251     "Mobile",
252     "Workstation",
253     "Enterprise Server",
254     "SOHO Server",
255     "Appliance PC",
256     "Performance Server",
257     "Tablet",
```

```
258     "Unknown Profile Type"
259 };

261 #define ACPI_GAS_WIDTH_RESERVED     5

263 static const char           *AcpiDmGasAccessWidth[] =
264 {
265     "Undefined/Legacy",
266     "Byte Access:8",
267     "Word Access:16",
268     "DWord Access:32",
269     "QWord Access:64",
270     "Unknown Width Encoding"
271 };


274 /*******************************************************************************
275  *
276  * ACPI Table Data, indexed by signature.
277  *
278  * Each entry contains: Signature, Table Info, Handler, DtHandler,
279  *  Template, Description
280  *
281  * Simple tables have only a TableInfo structure, complex tables have a
282  * handler. This table must be NULL terminated. RSDP and FACS are
283  * special-cased elsewhere.
284  *
285  ******************************************************************************/

287 ACPI_DMTABLE_DATA    AcpiDmTableData[] =
288 {
289     {ACPI_SIG_ASF,  NULL,                   AcpiDmDumpAsf,  DtCompileAsf,   Temp
290     {ACPI_SIG_BERT, AcpiDmTableInfoBert,    NULL,           NULL,           Temp
291     {ACPI_SIG_BGRT, AcpiDmTableInfoBgrt,    NULL,           NULL,           Temp
292     {ACPI_SIG_BOOT, AcpiDmTableInfoBoot,    NULL,           NULL,           Temp
293     {ACPI_SIG_CPEP, NULL,                   AcpiDmDumpCpep, DtCompileCpep,  Temp
294     {ACPI_SIG_CSRT, NULL,                   AcpiDmDumpCsrt, DtCompileCsrt,  Temp
295     {ACPI_SIG_DBG2, AcpiDmTableInfoDbg2,    AcpiDmDumpDbg2, DtCompileDbg2,  Temp
296     {ACPI_SIG_DBGP, AcpiDmTableInfoDbgp,    NULL,           NULL,           Temp
297     {ACPI_SIG_DMAR, NULL,                   AcpiDmDumpDmar, DtCompileDmar,  Temp
298     {ACPI_SIG_ECDT, AcpiDmTableInfoEcdt,    NULL,           NULL,           Temp
299     {ACPI_SIG_EINJ, NULL,                   AcpiDmDumpEinj, DtCompileEinj,  Temp
300     {ACPI_SIG_ERST, NULL,                   AcpiDmDumpErst, DtCompileErst,  Temp
301     {ACPI_SIG_FADT, NULL,                   AcpiDmDumpFadt, DtCompileFadt,  Temp
302     {ACPI_SIG_FPDT, NULL,                   AcpiDmDumpFpdt, DtCompileFpdt,  Temp
303     {ACPI_SIG_GTDT, AcpiDmTableInfoGtdt,    NULL,           NULL,           Temp
304     {ACPI_SIG_HEST, NULL,                   AcpiDmDumpHest, DtCompileHest,  Temp
305     {ACPI_SIG_HPET, AcpiDmTableInfoHpet,    NULL,           NULL,           Temp
306     {ACPI_SIG_IVRS, NULL,                   AcpiDmDumpIvrs, DtCompileIvrs,  Temp
307     {ACPI_SIG_MADT, NULL,                   AcpiDmDumpMadt, DtCompileMadt,  Temp
308     {ACPI_SIG_MCFG, NULL,                   AcpiDmDumpMcfg, DtCompileMcfg,  Temp
309     {ACPI_SIG_MCHI, AcpiDmTableInfoMchi,    NULL,           NULL,           Temp
310     {ACPI_SIG_MPST, AcpiDmTableInfoMpst,    AcpiDmDumpMpst, DtCompileMpst,  Temp
311     {ACPI_SIG_MSCT, NULL,                   AcpiDmDumpMsct, DtCompileMsct,  Temp
312     {ACPI_SIG_MTMR, NULL,                   AcpiDmDumpMtmr, DtCompileMtmr,  Temp
313     {ACPI_SIG_PCCT, AcpiDmTableInfoPcct,    AcpiDmDumpPcct, DtCompilePcct,  Temp
314     {ACPI_SIG_PMTT, NULL,                   AcpiDmDumpPmtt, DtCompilePmtt,  Temp
315     {ACPI_SIG_RSDT, NULL,                   AcpiDmDumpRsdt, DtCompileRsdt,  Temp
316     {ACPI_SIG_S3PT, NULL,                   NULL,           NULL,           Temp
317     {ACPI_SIG_SBST, AcpiDmTableInfoSbst,    NULL,           NULL,           Temp
318     {ACPI_SIG_SLIC, NULL,                   AcpiDmDumpSlic, DtCompileSlic,  Temp
319     {ACPI_SIG_SLIT, NULL,                   AcpiDmDumpSlit, DtCompileSlit,  Temp
320     {ACPI_SIG_SPCR, AcpiDmTableInfoSpcr,    NULL,           NULL,           Temp
321     {ACPI_SIG_SPMI, AcpiDmTableInfoSpmi,    NULL,           NULL,           Temp
322     {ACPI_SIG_SRAT, NULL,                   AcpiDmDumpSrat, DtCompileSrat,  Temp
323     {ACPI_SIG_TCPA, AcpiDmTableInfoTcpa,    NULL,           NULL,           Temp
```



```
324     {ACPI_SIG_TPM2, AcpiDmTableInfoTpm2,    NULL,           NULL,           Temp
325     {ACPI_SIG_UEFI, AcpiDmTableInfoUefi,    NULL,           DtCompileUefi,  Temp
326     {ACPI_SIG_VRTC, AcpiDmTableInfoVrtc,    AcpiDmDumpVrtc, DtCompileVrtc,  Temp
327     {ACPI_SIG_WAET, AcpiDmTableInfoWaet,    NULL,           NULL,           Temp
328     {ACPI_SIG_WDAT, NULL,                   AcpiDmDumpWdat, DtCompileWdat,  Temp
329     {ACPI_SIG_WDDT, AcpiDmTableInfoWddt,    NULL,           NULL,           Temp
330     {ACPI_SIG_WDRT, AcpiDmTableInfoWdrt,    NULL,           NULL,           Temp
331     {ACPI_SIG_XSDT, NULL,                   AcpiDmDumpXsdt, DtCompileXsdt,  Temp
332     {NULL,          NULL,                   NULL,           NULL,           NULL
333 };


336 /*******************************************************************************
337  *
338  * FUNCTION:    AcpiDmGenerateChecksum
339  *
340  * PARAMETERS:  Table               - Pointer to table to be checksummed
341  *              Length              - Length of the table
342  *              OriginalChecksum    - Value of the checksum field
343  *
344  * RETURN:      8 bit checksum of buffer
345  *
346  * DESCRIPTION: Computes an 8 bit checksum of the table.
347  *
348  ******************************************************************************/

350 UINT8
351 AcpiDmGenerateChecksum (
352     void                    *Table,
353     UINT32                  Length,
354     UINT8                   OriginalChecksum)
355 {
356     UINT8                   Checksum;


359     /* Sum the entire table as-is */

361     Checksum = AcpiTbChecksum ((UINT8 *) Table, Length);

363     /* Subtract off the existing checksum value in the table */

365     Checksum = (UINT8) (Checksum - OriginalChecksum);

367     /* Compute the final checksum */

369     Checksum = (UINT8) (0 - Checksum);
370     return (Checksum);
371 }


374 /*******************************************************************************
375  *
376  * FUNCTION:    AcpiDmGetTableData
377  *
378  * PARAMETERS:  Signature           - ACPI signature (4 chars) to match
379  *
380  * RETURN:      Pointer to a valid ACPI_DMTABLE_DATA. Null if no match found.
381  *
382  * DESCRIPTION: Find a match in the global table of supported ACPI tables
383  *
384  ******************************************************************************/

386 ACPI_DMTABLE_DATA *
387 AcpiDmGetTableData (
388     char                    *Signature)
389 {
```

```
390     ACPI_DMTABLE_DATA       *TableData;


393     for (TableData = AcpiDmTableData; TableData->Signature; TableData++)
394     {
395         if (ACPI_COMPARE_NAME (Signature, TableData->Signature))
396         {
397             return (TableData);
398         }
399     }

401     return (NULL);
402 }


405 /******************************************************************************
406  *
407  * FUNCTION:    AcpiDmDumpDataTable
408  *
409  * PARAMETERS:  Table               - An ACPI table
410  *
411  * RETURN:      None.
412  *
413  * DESCRIPTION: Format the contents of an ACPI data table (any table other
414  *              than an SSDT or DSDT that does not contain executable AML code)
415  *
416  *****************************************************************************/

418 void
419 AcpiDmDumpDataTable (
420     ACPI_TABLE_HEADER       *Table)
421 {
422     ACPI_STATUS             Status;
423     ACPI_DMTABLE_DATA       *TableData;
424     UINT32                  Length;


427     /* Ignore tables that contain AML */

429     if (AcpiUtIsAmlTable (Table))
430     {
431         if (Gbl_VerboseTemplates)
432         {
433             /* Dump the raw table data */

435             Length = Table->Length;

437             AcpiOsPrintf ("\n/*\n%s: Length %d (0x%X)\n\n",
438                 ACPI_RAW_TABLE_DATA_HEADER, Length, Length);
439             AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, Table),
440                 Length, DB_BYTE_DISPLAY, 0);
441             AcpiOsPrintf (" */\n");
442         }
443         return;
444     }

446     /*
447      * Handle tables that don't use the common ACPI table header structure.
448      * Currently, these are the FACS, RSDP, and S3PT.
449      */
450     if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_FACS))
451     {
452         Length = Table->Length;
453         AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoFacs);
454     }
455     else if (ACPI_VALIDATE_RSDP_SIG (Table->Signature))
```



```
456     {
457         Length = AcpiDmDumpRsdp (Table);
458     }
459     else if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_S3PT))
460     {
461         Length = AcpiDmDumpS3pt (Table);
462     }
463     else
464     {
465         /*
466          * All other tables must use the common ACPI table header, dump it now
467          */
468         Length = Table->Length;
469         Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoHeader);
470         if (ACPI_FAILURE (Status))
471         {
472             return;
473         }
474         AcpiOsPrintf ("\n");

476         /* Match signature and dispatch appropriately */

478         TableData = AcpiDmGetTableData (Table->Signature);
479         if (!TableData)
480         {
481             if (!ACPI_STRNCMP (Table->Signature, "OEM", 3))
482             {
483                 AcpiOsPrintf ("\n**** OEM-defined ACPI table [%4.4s], unknown co
484                     Table->Signature);
485             }
486             else
487             {
488                 AcpiOsPrintf ("\n**** Unknown ACPI table type [%4.4s]\n\n",
489                     Table->Signature);
490                 fprintf (stderr, "Unknown ACPI table signature [%4.4s], decoding
491                     Table->Signature);
492             }
493         }
494         else if (TableData->TableHandler)
495         {
496             /* Complex table, has a handler */

498             TableData->TableHandler (Table);
499         }
500         else if (TableData->TableInfo)
501         {
502             /* Simple table, just walk the info table */

504             AcpiDmDumpTable (Length, 0, Table, 0, TableData->TableInfo);
505         }
506     }

508     if (!Gbl_DoTemplates || Gbl_VerboseTemplates)
509     {
510         /* Dump the raw table data */

512         AcpiOsPrintf ("\n%s: Length %d (0x%X)\n\n",
513             ACPI_RAW_TABLE_DATA_HEADER, Length, Length);
514         AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, Table),
515             Length, DB_BYTE_DISPLAY, 0);
516     }
517 }


520 /******************************************************************************
521  *
```

```
 * FUNCTION:    AcpiDmLineHeader
 *
 * PARAMETERS:  Offset              - Current byte offset, from table start
 *              ByteLength          - Length of the field in bytes, 0 for flags
 *              Name                - Name of this field
 *              Value               - Optional value, displayed on left of ':'
 *
 * RETURN:      None
 *
 * DESCRIPTION: Utility routines for formatting output lines. Displays the
 *              current table offset in hex and decimal, the field length,
 *              and the field name.
 *
 *****************************************************************************/

void
AcpiDmLineHeader (
    UINT32                  Offset,
    UINT32                  ByteLength,
    char                    *Name)
{

    /* Allow a null name for fields that span multiple lines (large buffers) */

    if (!Name)
    {
        Name = "";
    }

    if (Gbl_DoTemplates && !Gbl_VerboseTemplates) /* Terse template */
    {
        if (ByteLength)
        {
            AcpiOsPrintf ("[%.4d] %34s : ", ByteLength, Name);
        }
        else
        {
            if (*Name)
            {
                AcpiOsPrintf ("%41s : ", Name);
            }
            else
            {
                AcpiOsPrintf ("%41s   ", Name);
            }
        }
    }
    else /* Normal disassembler or verbose template */
    {
        if (ByteLength)
        {
            AcpiOsPrintf ("[%3.3Xh %4.4d% 4d] %28s : ",
                Offset, Offset, ByteLength, Name);
        }
        else
        {
            if (*Name)
            {
                AcpiOsPrintf ("%44s : ", Name);
            }
            else
            {
                AcpiOsPrintf ("%44s   ", Name);
            }
        }
    }
```



```
}

void
AcpiDmLineHeader2 (
    UINT32                  Offset,
    UINT32                  ByteLength,
    char                    *Name,
    UINT32                  Value)
{

    if (Gbl_DoTemplates && !Gbl_VerboseTemplates) /* Terse template */
    {
        if (ByteLength)
        {
            AcpiOsPrintf ("[%.4d] %30s %3d : ",
                ByteLength, Name, Value);
        }
        else
        {
            AcpiOsPrintf ("%36s % 3d : ",
                Name, Value);
        }
    }
    else /* Normal disassembler or verbose template */
    {
        if (ByteLength)
        {
            AcpiOsPrintf ("[%3.3Xh %4.4d %3d] %24s %3d : ",
                Offset, Offset, ByteLength, Name, Value);
        }
        else
        {
            AcpiOsPrintf ("[%3.3Xh %4.4d   ] %24s %3d : ",
                Offset, Offset, Name, Value);
        }
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpTable
 *
 * PARAMETERS:  TableLength         - Length of the entire ACPI table
 *              TableOffset         - Starting offset within the table for this
 *                                    sub-descriptor (0 if main table)
 *              Table               - The ACPI table
 *              SubtableLength      - Length of this sub-descriptor
 *              Info                - Info table for this ACPI table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display ACPI table contents by walking the Info table.
 *
 * Note: This function must remain in sync with DtGetFieldLength.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiDmDumpTable (
    UINT32                  TableLength,
    UINT32                  TableOffset,
    void                    *Table,
    UINT32                  SubtableLength,
    ACPI_DMTABLE_INFO       *Info)
{
```

```
    UINT8                   *Target;
    UINT32                  CurrentOffset;
    UINT32                  ByteLength;
    UINT8                   Temp8;
    UINT16                  Temp16;
    UINT64                  Value;
    ACPI_DMTABLE_DATA       *TableData;
    const char              *Name;
    BOOLEAN                 LastOutputBlankLine = FALSE;
    char                    RepairedName[8];


    if (!Info)
    {
        AcpiOsPrintf ("Display not implemented\n");
        return (AE_NOT_IMPLEMENTED);
    }

    /* Walk entire Info table; Null name terminates */

    for (; Info->Name; Info++)
    {
        /*
         * Target points to the field within the ACPI Table. CurrentOffset is
         * the offset of the field from the start of the main table.
         */
        Target = ACPI_ADD_PTR (UINT8, Table, Info->Offset);
        CurrentOffset = TableOffset + Info->Offset;

        /* Check for beyond EOT or beyond subtable end */

        if ((CurrentOffset >= TableLength) ||
            (SubtableLength && (Info->Offset >= SubtableLength)))
        {
            AcpiOsPrintf ("**** ACPI table terminates in the middle of a data st
            return (AE_BAD_DATA);
        }

        /* Generate the byte length for this field */

        switch (Info->Opcode)
        {
        case ACPI_DMT_UINT8:
        case ACPI_DMT_CHKSUM:
        case ACPI_DMT_SPACEID:
        case ACPI_DMT_ACCWIDTH:
        case ACPI_DMT_IVRS:
        case ACPI_DMT_MADT:
        case ACPI_DMT_PCCT:
        case ACPI_DMT_PMTT:
        case ACPI_DMT_SRAT:
        case ACPI_DMT_ASF:
        case ACPI_DMT_HESTNTYP:
        case ACPI_DMT_FADTPM:
        case ACPI_DMT_EINJACT:
        case ACPI_DMT_EINJINST:
        case ACPI_DMT_ERSTACT:
        case ACPI_DMT_ERSTINST:

            ByteLength = 1;
            break;

        case ACPI_DMT_UINT16:
        case ACPI_DMT_DMAR:
        case ACPI_DMT_HEST:
```



```
            ByteLength = 2;
            break;

        case ACPI_DMT_UINT24:

            ByteLength = 3;
            break;

        case ACPI_DMT_UINT32:
        case ACPI_DMT_NAME4:
        case ACPI_DMT_SIG:
        case ACPI_DMT_SLIC:

            ByteLength = 4;
            break;

        case ACPI_DMT_UINT40:

            ByteLength = 5;
            break;

        case ACPI_DMT_UINT48:
        case ACPI_DMT_NAME6:

            ByteLength = 6;
            break;

        case ACPI_DMT_UINT56:
        case ACPI_DMT_BUF7:

            ByteLength = 7;
            break;

        case ACPI_DMT_UINT64:
        case ACPI_DMT_NAME8:

            ByteLength = 8;
            break;

        case ACPI_DMT_BUF10:

            ByteLength = 10;
            break;

        case ACPI_DMT_BUF16:
        case ACPI_DMT_UUID:

            ByteLength = 16;
            break;

        case ACPI_DMT_BUF128:

            ByteLength = 128;
            break;

        case ACPI_DMT_STRING:

            ByteLength = ACPI_STRLEN (ACPI_CAST_PTR (char, Target)) + 1;
            break;

        case ACPI_DMT_GAS:

            if (!LastOutputBlankLine)
            {
                AcpiOsPrintf ("\n");
                LastOutputBlankLine = TRUE;
```

```
786             }
787             ByteLength = sizeof (ACPI_GENERIC_ADDRESS);
788             break;

790         case ACPI_DMT_HESTNTFY:

792             if (!LastOutputBlankLine)
793             {
794                 AcpiOsPrintf ("\n");
795                 LastOutputBlankLine = TRUE;
796             }
797             ByteLength = sizeof (ACPI_HEST_NOTIFY);
798             break;

800         default:

802             ByteLength = 0;
803             break;
804         }

806         if (CurrentOffset + ByteLength > TableLength)
807         {
808             AcpiOsPrintf ("**** ACPI table terminates in the middle of a data st
809             return (AE_BAD_DATA);
810         }

812         if (Info->Opcode == ACPI_DMT_EXTRA_TEXT)
813         {
814             AcpiOsPrintf ("%s", Info->Name);
815             continue;
816         }

818         /* Start a new line and decode the opcode */

820         AcpiDmLineHeader (CurrentOffset, ByteLength, Info->Name);

822         switch (Info->Opcode)
823         {
824         /* Single-bit Flag fields. Note: Opcode is the bit position */

826         case ACPI_DMT_FLAG0:
827         case ACPI_DMT_FLAG1:
828         case ACPI_DMT_FLAG2:
829         case ACPI_DMT_FLAG3:
830         case ACPI_DMT_FLAG4:
831         case ACPI_DMT_FLAG5:
832         case ACPI_DMT_FLAG6:
833         case ACPI_DMT_FLAG7:

835             AcpiOsPrintf ("%1.1X\n", (*Target >> Info->Opcode) & 0x01);
836             break;

838         /* 2-bit Flag fields */

840         case ACPI_DMT_FLAGS0:

842             AcpiOsPrintf ("%1.1X\n", *Target & 0x03);
843             break;

845         case ACPI_DMT_FLAGS1:

847             AcpiOsPrintf ("%1.1X\n", (*Target >> 1) & 0x03);
848             break;

850         case ACPI_DMT_FLAGS2:
```



```
852             AcpiOsPrintf ("%1.1X\n", (*Target >> 2) & 0x03);
853             break;

855         case ACPI_DMT_FLAGS4:

857             AcpiOsPrintf ("%1.1X\n", (*Target >> 4) & 0x03);
858             break;

860         /* Integer Data Types */

862         case ACPI_DMT_UINT8:
863         case ACPI_DMT_UINT16:
864         case ACPI_DMT_UINT24:
865         case ACPI_DMT_UINT32:
866         case ACPI_DMT_UINT40:
867         case ACPI_DMT_UINT48:
868         case ACPI_DMT_UINT56:
869         case ACPI_DMT_UINT64:
870             /*
871              * Dump bytes - high byte first, low byte last.
872              * Note: All ACPI tables are little-endian.
873              */
874             Value = 0;
875             for (Temp8 = (UINT8) ByteLength; Temp8 > 0; Temp8--)
876             {
877                 AcpiOsPrintf ("%2.2X", Target[Temp8 - 1]);
878                 Value |= Target[Temp8 - 1];
879                 Value <<= 8;
880             }

882             if (!Value && (Info->Flags & DT_DESCRIBES_OPTIONAL))
883             {
884                 AcpiOsPrintf (" [Optional field not present]");
885             }

887             AcpiOsPrintf ("\n");
888             break;

890         case ACPI_DMT_BUF7:
891         case ACPI_DMT_BUF10:
892         case ACPI_DMT_BUF16:
893         case ACPI_DMT_BUF128:
894             /*
895              * Buffer: Size depends on the opcode and was set above.
896              * Each hex byte is separated with a space.
897              * Multiple lines are separated by line continuation char.
898              */
899             for (Temp16 = 0; Temp16 < ByteLength; Temp16++)
900             {
901                 AcpiOsPrintf ("%2.2X", Target[Temp16]);
902                 if ((UINT32) (Temp16 + 1) < ByteLength)
903                 {
904                     if ((Temp16 > 0) && (!((Temp16+1) % 16)))
905                     {
906                         AcpiOsPrintf (" \\\n"); /* Line continuation */
907                         AcpiDmLineHeader (0, 0, NULL);
908                     }
909                     else
910                     {
911                         AcpiOsPrintf (" ");
912                     }
913                 }
914             }
915             AcpiOsPrintf ("\n");
916             break;
```

```
918         case ACPI_DMT_UUID:
920             /* Convert 16-byte UUID buffer to 36-byte formatted UUID string */
922             (void) AuConvertUuidToString ((char *) Target, MsgBuffer);
924             AcpiOsPrintf ("%s\n", MsgBuffer);
925             break;
927         case ACPI_DMT_STRING:
929             AcpiOsPrintf ("\"%s\"\n", ACPI_CAST_PTR (char, Target));
930             break;
932         /* Fixed length ASCII name fields */
934         case ACPI_DMT_SIG:
936             AcpiDmCheckAscii (Target, RepairedName, 4);
937             AcpiOsPrintf ("\"%.4s\"    ", RepairedName);
938             TableData = AcpiDmGetTableData (ACPI_CAST_PTR (char, Target));
939             if (TableData)
940             {
941                 AcpiOsPrintf (STRING_FORMAT, TableData->Name);
942             }
943             else
944             {
945                 AcpiOsPrintf ("\n");
946             }
947             break;
949         case ACPI_DMT_NAME4:
951             AcpiDmCheckAscii (Target, RepairedName, 4);
952             AcpiOsPrintf ("\"%.4s\"\n", RepairedName);
953             break;
955         case ACPI_DMT_NAME6:
957             AcpiDmCheckAscii (Target, RepairedName, 6);
958             AcpiOsPrintf ("\"%.6s\"\n", RepairedName);
959             break;
961         case ACPI_DMT_NAME8:
963             AcpiDmCheckAscii (Target, RepairedName, 8);
964             AcpiOsPrintf ("\"%.8s\"\n", RepairedName);
965             break;
967         /* Special Data Types */
969         case ACPI_DMT_CHKSUM:
971             /* Checksum, display and validate */
973             AcpiOsPrintf ("%2.2X", *Target);
974             Temp8 = AcpiDmGenerateChecksum (Table,
975                         ACPI_CAST_PTR (ACPI_TABLE_HEADER, Table)->Length,
976                         ACPI_CAST_PTR (ACPI_TABLE_HEADER, Table)->Checksum);
977             if (Temp8 != ACPI_CAST_PTR (ACPI_TABLE_HEADER, Table)->Checksum)
978             {
979                 AcpiOsPrintf (
980                     "     /* Incorrect checksum, should be %2.2X */", Temp8);
981             }
982             AcpiOsPrintf ("\n");
983             break;
```



```
985         case ACPI_DMT_SPACEID:
987             /* Address Space ID */
989             AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiUtGetRegionName (*Target));
990             break;
992         case ACPI_DMT_ACCWIDTH:
994             /* Encoded Access Width */
996             Temp8 = *Target;
997             if (Temp8 > ACPI_GAS_WIDTH_RESERVED)
998             {
999                 Temp8 = ACPI_GAS_WIDTH_RESERVED;
1000            }
1002            AcpiOsPrintf (UINT8_FORMAT, Temp8, AcpiDmGasAccessWidth[Temp8]);
1003            break;
1005        case ACPI_DMT_GAS:
1007            /* Generic Address Structure */
1009            AcpiOsPrintf (STRING_FORMAT, "Generic Address Structure");
1010            AcpiDmDumpTable (TableLength, CurrentOffset, Target,
1011                sizeof (ACPI_GENERIC_ADDRESS), AcpiDmTableInfoGas);
1012            AcpiOsPrintf ("\n");
1013            LastOutputBlankLine = TRUE;
1014            break;
1016        case ACPI_DMT_ASF:
1018            /* ASF subtable types */
1020            Temp16 = (UINT16) ((*Target) & 0x7F);  /* Top bit can be zero or one
1021            if (Temp16 > ACPI_ASF_TYPE_RESERVED)
1022            {
1023                Temp16 = ACPI_ASF_TYPE_RESERVED;
1024            }
1026            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmAsfSubnames[Temp16]);
1027            break;
1029        case ACPI_DMT_DMAR:
1031            /* DMAR subtable types */
1033            Temp16 = ACPI_GET16 (Target);
1034            if (Temp16 > ACPI_DMAR_TYPE_RESERVED)
1035            {
1036                Temp16 = ACPI_DMAR_TYPE_RESERVED;
1037            }
1039            AcpiOsPrintf (UINT16_FORMAT, ACPI_GET16 (Target), AcpiDmDmarSubnames
1040            break;
1042        case ACPI_DMT_EINJACT:
1044            /* EINJ Action types */
1046            Temp8 = *Target;
1047            if (Temp8 > ACPI_EINJ_ACTION_RESERVED)
1048            {
1049                Temp8 = ACPI_EINJ_ACTION_RESERVED;
```

```
1050            }
1052            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmEinjActions[Temp8]);
1053            break;
1055        case ACPI_DMT_EINJINST:
1057            /* EINJ Instruction types */
1059            Temp8 = *Target;
1060            if (Temp8 > ACPI_EINJ_INSTRUCTION_RESERVED)
1061            {
1062                Temp8 = ACPI_EINJ_INSTRUCTION_RESERVED;
1063            }
1065            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmEinjInstructions[Temp8]);
1066            break;
1068        case ACPI_DMT_ERSTACT:
1070            /* ERST Action types */
1072            Temp8 = *Target;
1073            if (Temp8 > ACPI_ERST_ACTION_RESERVED)
1074            {
1075                Temp8 = ACPI_ERST_ACTION_RESERVED;
1076            }
1078            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmErstActions[Temp8]);
1079            break;
1081        case ACPI_DMT_ERSTINST:
1083            /* ERST Instruction types */
1085            Temp8 = *Target;
1086            if (Temp8 > ACPI_ERST_INSTRUCTION_RESERVED)
1087            {
1088                Temp8 = ACPI_ERST_INSTRUCTION_RESERVED;
1089            }
1091            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmErstInstructions[Temp8]);
1092            break;
1094        case ACPI_DMT_HEST:
1096            /* HEST subtable types */
1098            Temp16 = ACPI_GET16 (Target);
1099            if (Temp16 > ACPI_HEST_TYPE_RESERVED)
1100            {
1101                Temp16 = ACPI_HEST_TYPE_RESERVED;
1102            }
1104            AcpiOsPrintf (UINT16_FORMAT, ACPI_GET16 (Target), AcpiDmHestSubnames
1105            break;
1107        case ACPI_DMT_HESTNTFY:
1109            AcpiOsPrintf (STRING_FORMAT, "Hardware Error Notification Structure"
1110            AcpiDmDumpTable (TableLength, CurrentOffset, Target,
1111                sizeof (ACPI_HEST_NOTIFY), AcpiDmTableInfoHestNotify);
1112            AcpiOsPrintf ("\n");
1113            LastOutputBlankLine = TRUE;
1114            break;
```



```
1116        case ACPI_DMT_HESTNTYP:
1118            /* HEST Notify types */
1120            Temp8 = *Target;
1121            if (Temp8 > ACPI_HEST_NOTIFY_RESERVED)
1122            {
1123                Temp8 = ACPI_HEST_NOTIFY_RESERVED;
1124            }
1126            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmHestNotifySubnames[Temp8]
1127            break;
1129        case ACPI_DMT_MADT:
1131            /* MADT subtable types */
1133            Temp8 = *Target;
1134            if (Temp8 > ACPI_MADT_TYPE_RESERVED)
1135            {
1136                Temp8 = ACPI_MADT_TYPE_RESERVED;
1137            }
1139            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmMadtSubnames[Temp8]);
1140            break;
1142        case ACPI_DMT_PCCT:
1144            /* PCCT subtable types */
1146            Temp8 = *Target;
1147            if (Temp8 > ACPI_PCCT_TYPE_RESERVED)
1148            {
1149                Temp8 = ACPI_PCCT_TYPE_RESERVED;
1150            }
1152            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmPcctSubnames[Temp8]);
1153            break;
1155        case ACPI_DMT_PMTT:
1157            /* PMTT subtable types */
1159            Temp8 = *Target;
1160            if (Temp8 > ACPI_PMTT_TYPE_RESERVED)
1161            {
1162                Temp8 = ACPI_PMTT_TYPE_RESERVED;
1163            }
1165            AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmPmttSubnames[Temp8]);
1166            break;
1168        case ACPI_DMT_SLIC:
1170            /* SLIC subtable types */
1172            Temp8 = *Target;
1173            if (Temp8 > ACPI_SLIC_TYPE_RESERVED)
1174            {
1175                Temp8 = ACPI_SLIC_TYPE_RESERVED;
1176            }
1178            AcpiOsPrintf (UINT32_FORMAT, *Target, AcpiDmSlicSubnames[Temp8]);
1179            break;
1181        case ACPI_DMT_SRAT:
```

```
1183             /* SRAT subtable types */
1185             Temp8 = *Target;
1186             if (Temp8 > ACPI_SRAT_TYPE_RESERVED)
1187             {
1188                 Temp8 = ACPI_SRAT_TYPE_RESERVED;
1189             }
1191             AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmSratSubnames[Temp8]);
1192             break;
1194         case ACPI_DMT_FADTPM:
1196             /* FADT Preferred PM Profile names */
1198             Temp8 = *Target;
1199             if (Temp8 > ACPI_FADT_PM_RESERVED)
1200             {
1201                 Temp8 = ACPI_FADT_PM_RESERVED;
1202             }
1204             AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmFadtProfiles[Temp8]);
1205             break;
1207         case ACPI_DMT_IVRS:
1209             /* IVRS subtable types */
1211             Temp8 = *Target;
1212             switch (Temp8)
1213             {
1214             case ACPI_IVRS_TYPE_HARDWARE:
1216                 Name = AcpiDmIvrsSubnames[0];
1217                 break;
1219             case ACPI_IVRS_TYPE_MEMORY1:
1220             case ACPI_IVRS_TYPE_MEMORY2:
1221             case ACPI_IVRS_TYPE_MEMORY3:
1223                 Name = AcpiDmIvrsSubnames[1];
1224                 break;
1226             default:
1228                 Name = AcpiDmIvrsSubnames[2];
1229                 break;
1230             }
1232             AcpiOsPrintf (UINT8_FORMAT, *Target, Name);
1233             break;
1235         case ACPI_DMT_EXIT:
1237             return (AE_OK);
1239         default:
1241             ACPI_ERROR ((AE_INFO,
1242                 "**** Invalid table opcode [0x%X] ****\n", Info->Opcode));
1243             return (AE_SUPPORT);
1244         }
1245     }
1247     if (TableOffset && !SubtableLength)
```



```
1248     {
1249         /* If this table is not the main table, subtable must have valid length
1251         AcpiOsPrintf ("Invalid zero length subtable\n");
1252         return (AE_BAD_DATA);
1253     }
1255     return (AE_OK);
1256 }


1259 /******************************************************************************
1260  *
1261  * FUNCTION:    AcpiDmCheckAscii
1262  *
1263  * PARAMETERS:  Name                - Ascii string
1264  *              Count               - Number of characters to check
1265  *
1266  * RETURN:      None
1267  *
1268  * DESCRIPTION: Ensure that the requested number of characters are printable
1269  *              Ascii characters. Sets non-printable and null chars to <space>.
1270  *
1271  *****************************************************************************/

1273 static void
1274 AcpiDmCheckAscii (
1275     UINT8                   *Name,
1276     char                    *RepairedName,
1277     UINT32                  Count)
1278 {
1279     UINT32                  i;


1282     for (i = 0; i < Count; i++)
1283     {
1284         RepairedName[i] = (char) Name[i];
1286         if (!Name[i])
1287         {
1288             return;
1289         }
1290         if (!isprint (Name[i]))
1291         {
1292             RepairedName[i] = ' ';
1293         }
1294     }
1295 }
```

```
**********************************************************
   73554 Thu Dec 26 13:48:24 2013
new/usr/src/common/acpica/common/dmtbdump.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmtbdump - Dump ACPI data tables that contain no AML code
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"
#include "actables.h"

/* This module used for application-level code only */

#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmtbdump")


/* Local prototypes */

static void
AcpiDmValidateFadtLength (
    UINT32                  Revision,
    UINT32                  Length);

static void
AcpiDmDumpBuffer (
    void                    *Table,
    UINT32                  BufferOffset,
    UINT32                  Length,
    UINT32                  AbsoluteOffset,
    char                    *Header);


/******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpBuffer
 *
 * PARAMETERS:  Table               - ACPI Table or subtable
 *              BufferOffset        - Offset of buffer from Table above
 *              Length              - Length of the buffer
 *              AbsoluteOffset      - Offset of buffer in the main ACPI table
 *              Header              - Name of the buffer field (printed on the
 *                                    first line only.)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of an arbitrary length data buffer (in the
 *              disassembler output format.)
 *
 *****************************************************************************/

static void
AcpiDmDumpBuffer (
    void                    *Table,
    UINT32                  BufferOffset,
    UINT32                  Length,
    UINT32                  AbsoluteOffset,
    char                    *Header)
{
    UINT8                   *Buffer;
    UINT32                  i;


    if (!Length)
    {
        return;
    }

    Buffer = ACPI_CAST_PTR (UINT8, Table) + BufferOffset;
    i = 0;

    while (i < Length)
    {
        if (!(i % 16))
        {
            AcpiOsPrintf ("\n");
            AcpiDmLineHeader (AbsoluteOffset,
                ((Length - i) > 16) ? 16 : (Length - i), Header);
            Header = NULL;
        }

        AcpiOsPrintf ("%.02X ", *Buffer);
        i++;
        Buffer++;
        AbsoluteOffset++;
    }

    AcpiOsPrintf ("\n");
```

```
126 }


129 /*******************************************************************************
130  *
131  * FUNCTION:    AcpiDmDumpRsdp
132  *
133  * PARAMETERS:  Table               - A RSDP
134  *
135  * RETURN:      Length of the table (there is not always a length field,
136  *              use revision or length if available (ACPI 2.0+))
137  *
138  * DESCRIPTION: Format the contents of a RSDP
139  *
140  ******************************************************************************/

142 UINT32
143 AcpiDmDumpRsdp (
144     ACPI_TABLE_HEADER       *Table)
145 {
146     ACPI_TABLE_RSDP         *Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, Table);
147     UINT32                  Length = sizeof (ACPI_RSDP_COMMON);
148     UINT8                   Checksum;


151     /* Dump the common ACPI 1.0 portion */

153     AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoRsdp1);

155     /* Validate the first checksum */

157     Checksum = AcpiDmGenerateChecksum (Rsdp, sizeof (ACPI_RSDP_COMMON),
158                 Rsdp->Checksum);
159     if (Checksum != Rsdp->Checksum)
160     {
161         AcpiOsPrintf ("/* Incorrect Checksum above, should be 0x%2.2X */\n",
162             Checksum);
163     }

165     /* The RSDP for ACPI 2.0+ contains more data and has a Length field */

167     if (Rsdp->Revision > 0)
168     {
169         Length = Rsdp->Length;
170         AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoRsdp2);

172         /* Validate the extended checksum over entire RSDP */

174         Checksum = AcpiDmGenerateChecksum (Rsdp, sizeof (ACPI_TABLE_RSDP),
175                     Rsdp->ExtendedChecksum);
176         if (Checksum != Rsdp->ExtendedChecksum)
177         {
178             AcpiOsPrintf (
179                 "/* Incorrect Extended Checksum above, should be 0x%2.2X */\n",
180                 Checksum);
181         }
182     }

184     return (Length);
185 }


188 /*******************************************************************************
189  *
190  * FUNCTION:    AcpiDmDumpRsdt
191  *
```



```
192  * PARAMETERS:  Table               - A RSDT
193  *
194  * RETURN:      None
195  *
196  * DESCRIPTION: Format the contents of a RSDT
197  *
198  ******************************************************************************/

200 void
201 AcpiDmDumpRsdt (
202     ACPI_TABLE_HEADER       *Table)
203 {
204     UINT32                  *Array;
205     UINT32                  Entries;
206     UINT32                  Offset;
207     UINT32                  i;


210     /* Point to start of table pointer array */

212     Array = ACPI_CAST_PTR (ACPI_TABLE_RSDT, Table)->TableOffsetEntry;
213     Offset = sizeof (ACPI_TABLE_HEADER);

215     /* RSDT uses 32-bit pointers */

217     Entries = (Table->Length - sizeof (ACPI_TABLE_HEADER)) / sizeof (UINT32);

219     for (i = 0; i < Entries; i++)
220     {
221         AcpiDmLineHeader2 (Offset, sizeof (UINT32), "ACPI Table Address", i);
222         AcpiOsPrintf ("%8.8X\n", Array[i]);
223         Offset += sizeof (UINT32);
224     }
225 }


228 /*******************************************************************************
229  *
230  * FUNCTION:    AcpiDmDumpXsdt
231  *
232  * PARAMETERS:  Table               - A XSDT
233  *
234  * RETURN:      None
235  *
236  * DESCRIPTION: Format the contents of a XSDT
237  *
238  ******************************************************************************/

240 void
241 AcpiDmDumpXsdt (
242     ACPI_TABLE_HEADER       *Table)
243 {
244     UINT64                  *Array;
245     UINT32                  Entries;
246     UINT32                  Offset;
247     UINT32                  i;


250     /* Point to start of table pointer array */

252     Array = ACPI_CAST_PTR (ACPI_TABLE_XSDT, Table)->TableOffsetEntry;
253     Offset = sizeof (ACPI_TABLE_HEADER);

255     /* XSDT uses 64-bit pointers */

257     Entries = (Table->Length - sizeof (ACPI_TABLE_HEADER)) / sizeof (UINT64);
```

```
259     for (i = 0; i < Entries; i++)
260     {
261         AcpiDmLineHeader2 (Offset, sizeof (UINT64), "ACPI Table Address", i);
262         AcpiOsPrintf ("%8.8X%8.8X\n", ACPI_FORMAT_UINT64 (Array[i]));
263         Offset += sizeof (UINT64);
264     }
265 }


268 /*******************************************************************************
269  *
270  * FUNCTION:    AcpiDmDumpFadt
271  *
272  * PARAMETERS:  Table               - A FADT
273  *
274  * RETURN:      None
275  *
276  * DESCRIPTION: Format the contents of a FADT
277  *
278  * NOTE:        We cannot depend on the FADT version to indicate the actual
279  *              contents of the FADT because of BIOS bugs. The table length
280  *              is the only reliable indicator.
281  *
282  ******************************************************************************/

284 void
285 AcpiDmDumpFadt (
286     ACPI_TABLE_HEADER       *Table)
287 {

289     /* Always dump the minimum FADT revision 1 fields (ACPI 1.0) */

291     AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt1);

293     /* Check for FADT revision 2 fields (ACPI 1.0B MS extensions) */

295     if ((Table->Length > ACPI_FADT_V1_SIZE) &&
296         (Table->Length <= ACPI_FADT_V2_SIZE))
297     {
298         AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt2);
299     }

301     /* Check for FADT revision 3/4 fields and up (ACPI 2.0+ extended data) */

303     else if (Table->Length > ACPI_FADT_V2_SIZE)
304     {
305         AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt3);

307         /* Check for FADT revision 5 fields and up (ACPI 5.0+) */

309         if (Table->Length > ACPI_FADT_V3_SIZE)
310         {
311             AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt5);
312         }
313     }

315     /* Validate various fields in the FADT, including length */

317     AcpiTbCreateLocalFadt (Table, Table->Length);

319     /* Validate FADT length against the revision */

321     AcpiDmValidateFadtLength (Table->Revision, Table->Length);
322 }
```



```
325 /*******************************************************************************
326  *
327  * FUNCTION:    AcpiDmValidateFadtLength
328  *
329  * PARAMETERS:  Revision            - FADT revision (Header->Revision)
330  *              Length              - FADT length (Header->Length
331  *
332  * RETURN:      None
333  *
334  * DESCRIPTION: Check the FADT revision against the expected table length for
335  *              that revision. Issue a warning if the length is not what was
336  *              expected. This seems to be such a common BIOS bug that the
337  *              FADT revision has been rendered virtually meaningless.
338  *
339  ******************************************************************************/

341 static void
342 AcpiDmValidateFadtLength (
343     UINT32                  Revision,
344     UINT32                  Length)
345 {
346     UINT32                  ExpectedLength;


349     switch (Revision)
350     {
351     case 0:

353         AcpiOsPrintf ("// ACPI Warning: Invalid FADT revision: 0\n");
354         return;

356     case 1:

358         ExpectedLength = ACPI_FADT_V1_SIZE;
359         break;

361     case 2:

363         ExpectedLength = ACPI_FADT_V2_SIZE;
364         break;

366     case 3:
367     case 4:

369         ExpectedLength = ACPI_FADT_V3_SIZE;
370         break;

372     case 5:

374         ExpectedLength = ACPI_FADT_V5_SIZE;
375         break;

377     default:

379         return;
380     }

382     if (Length == ExpectedLength)
383     {
384         return;
385     }

387     AcpiOsPrintf (
388         "\n// ACPI Warning: FADT revision %X does not match length: found %X exp
389         Revision, Length, ExpectedLength);
```

```
390 }


393 /*******************************************************************************
394  *
395  * FUNCTION:    AcpiDmDumpAsf
396  *
397  * PARAMETERS:  Table               - A ASF table
398  *
399  * RETURN:      None
400  *
401  * DESCRIPTION: Format the contents of a ASF table
402  *
403  ******************************************************************************/

405 void
406 AcpiDmDumpAsf (
407     ACPI_TABLE_HEADER       *Table)
408 {
409     ACPI_STATUS             Status;
410     UINT32                  Offset = sizeof (ACPI_TABLE_HEADER);
411     ACPI_ASF_INFO           *SubTable;
412     ACPI_DMTABLE_INFO       *InfoTable;
413     ACPI_DMTABLE_INFO       *DataInfoTable = NULL;
414     UINT8                   *DataTable = NULL;
415     UINT32                  DataCount = 0;
416     UINT32                  DataLength = 0;
417     UINT32                  DataOffset = 0;
418     UINT32                  i;
419     UINT8                   Type;


422     /* No main table, only sub-tables */

424     SubTable = ACPI_ADD_PTR (ACPI_ASF_INFO, Table, Offset);
425     while (Offset < Table->Length)
426     {
427         /* Common sub-table header */

429         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
430                     SubTable->Header.Length, AcpiDmTableInfoAsfHdr);
431         if (ACPI_FAILURE (Status))
432         {
433             return;
434         }

436         /* The actual type is the lower 7 bits of Type */

438         Type = (UINT8) (SubTable->Header.Type & 0x7F);

440         switch (Type)
441         {
442         case ACPI_ASF_TYPE_INFO:

444             InfoTable = AcpiDmTableInfoAsf0;
445             break;

447         case ACPI_ASF_TYPE_ALERT:

449             InfoTable = AcpiDmTableInfoAsf1;
450             DataInfoTable = AcpiDmTableInfoAsf1a;
451             DataTable = ACPI_ADD_PTR (UINT8, SubTable, sizeof (ACPI_ASF_ALERT));
452             DataCount = ACPI_CAST_PTR (ACPI_ASF_ALERT, SubTable)->Alerts;
453             DataLength = ACPI_CAST_PTR (ACPI_ASF_ALERT, SubTable)->DataLength;
454             DataOffset = Offset + sizeof (ACPI_ASF_ALERT);
455             break;
```



```
457         case ACPI_ASF_TYPE_CONTROL:

459             InfoTable = AcpiDmTableInfoAsf2;
460             DataInfoTable = AcpiDmTableInfoAsf2a;
461             DataTable = ACPI_ADD_PTR (UINT8, SubTable, sizeof (ACPI_ASF_REMOTE))
462             DataCount = ACPI_CAST_PTR (ACPI_ASF_REMOTE, SubTable)->Controls;
463             DataLength = ACPI_CAST_PTR (ACPI_ASF_REMOTE, SubTable)->DataLength;
464             DataOffset = Offset + sizeof (ACPI_ASF_REMOTE);
465             break;

467         case ACPI_ASF_TYPE_BOOT:

469             InfoTable = AcpiDmTableInfoAsf3;
470             break;

472         case ACPI_ASF_TYPE_ADDRESS:

474             InfoTable = AcpiDmTableInfoAsf4;
475             DataTable = ACPI_ADD_PTR (UINT8, SubTable, sizeof (ACPI_ASF_ADDRESS)
476             DataLength = ACPI_CAST_PTR (ACPI_ASF_ADDRESS, SubTable)->Devices;
477             DataOffset = Offset + sizeof (ACPI_ASF_ADDRESS);
478             break;

480         default:

482             AcpiOsPrintf ("\n**** Unknown ASF sub-table type 0x%X\n", SubTable->
483             return;
484         }

486         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
487                     SubTable->Header.Length, InfoTable);
488         if (ACPI_FAILURE (Status))
489         {
490             return;
491         }

493         /* Dump variable-length extra data */

495         switch (Type)
496         {
497         case ACPI_ASF_TYPE_ALERT:
498         case ACPI_ASF_TYPE_CONTROL:

500             for (i = 0; i < DataCount; i++)
501             {
502                 AcpiOsPrintf ("\n");
503                 Status = AcpiDmDumpTable (Table->Length, DataOffset,
504                             DataTable, DataLength, DataInfoTable);
505                 if (ACPI_FAILURE (Status))
506                 {
507                     return;
508                 }

510                 DataTable = ACPI_ADD_PTR (UINT8, DataTable, DataLength);
511                 DataOffset += DataLength;
512             }
513             break;

515         case ACPI_ASF_TYPE_ADDRESS:

517             for (i = 0; i < DataLength; i++)
518             {
519                 if (!(i % 16))
520                 {
521                     AcpiDmLineHeader (DataOffset, 1, "Addresses");
```

```
522                 }
524                 AcpiOsPrintf ("%2.2X ", *DataTable);
525                 DataTable++;
526                 DataOffset++;
527                 if (DataOffset > Table->Length)
528                 {
529                     AcpiOsPrintf ("**** ACPI table terminates in the middle of a
530                     return;
531                 }
532             }
534             AcpiOsPrintf ("\n");
535             break;
537         default:
539             break;
540         }
542         AcpiOsPrintf ("\n");
544         /* Point to next sub-table */
546         if (!SubTable->Header.Length)
547         {
548             AcpiOsPrintf ("Invalid zero subtable header length\n");
549             return;
550         }
552         Offset += SubTable->Header.Length;
553         SubTable = ACPI_ADD_PTR (ACPI_ASF_INFO, SubTable, SubTable->Header.Lengt
554     }
555 }
558 /*******************************************************************************
559  *
560  * FUNCTION:    AcpiDmDumpCpep
561  *
562  * PARAMETERS:  Table               - A CPEP table
563  *
564  * RETURN:      None
565  *
566  * DESCRIPTION: Format the contents of a CPEP. This table type consists
567  *              of an open-ended number of subtables.
568  *
569  ******************************************************************************/
571 void
572 AcpiDmDumpCpep (
573     ACPI_TABLE_HEADER       *Table)
574 {
575     ACPI_STATUS             Status;
576     ACPI_CPEP_POLLING       *SubTable;
577     UINT32                  Length = Table->Length;
578     UINT32                  Offset = sizeof (ACPI_TABLE_CPEP);
581     /* Main table */
583     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoCpep);
584     if (ACPI_FAILURE (Status))
585     {
586         return;
587     }
```



```
589     /* Sub-tables */
591     SubTable = ACPI_ADD_PTR (ACPI_CPEP_POLLING, Table, Offset);
592     while (Offset < Table->Length)
593     {
594         AcpiOsPrintf ("\n");
595         Status = AcpiDmDumpTable (Length, Offset, SubTable,
596                     SubTable->Header.Length, AcpiDmTableInfoCpep0);
597         if (ACPI_FAILURE (Status))
598         {
599             return;
600         }
602         /* Point to next sub-table */
604         Offset += SubTable->Header.Length;
605         SubTable = ACPI_ADD_PTR (ACPI_CPEP_POLLING, SubTable,
606                     SubTable->Header.Length);
607     }
608 }
611 /*******************************************************************************
612  *
613  * FUNCTION:    AcpiDmDumpCsrt
614  *
615  * PARAMETERS:  Table               - A CSRT table
616  *
617  * RETURN:      None
618  *
619  * DESCRIPTION: Format the contents of a CSRT. This table type consists
620  *              of an open-ended number of subtables.
621  *
622  ******************************************************************************/
624 void
625 AcpiDmDumpCsrt (
626     ACPI_TABLE_HEADER       *Table)
627 {
628     ACPI_STATUS             Status;
629     ACPI_CSRT_GROUP         *SubTable;
630     ACPI_CSRT_SHARED_INFO   *SharedInfoTable;
631     ACPI_CSRT_DESCRIPTOR    *SubSubTable;
632     UINT32                  Length = Table->Length;
633     UINT32                  Offset = sizeof (ACPI_TABLE_CSRT);
634     UINT32                  SubOffset;
635     UINT32                  SubSubOffset;
636     UINT32                  InfoLength;
639     /* The main table only contains the ACPI header, thus already handled */
641     /* Sub-tables (Resource Groups) */
643     SubTable = ACPI_ADD_PTR (ACPI_CSRT_GROUP, Table, Offset);
644     while (Offset < Table->Length)
645     {
646         /* Resource group subtable */
648         AcpiOsPrintf ("\n");
649         Status = AcpiDmDumpTable (Length, Offset, SubTable,
650                     SubTable->Length, AcpiDmTableInfoCsrt0);
651         if (ACPI_FAILURE (Status))
652         {
653             return;
```

```
654         }

656         /* Shared info subtable (One per resource group) */

658         SubOffset = sizeof (ACPI_CSRT_GROUP);
659         SharedInfoTable = ACPI_ADD_PTR (ACPI_CSRT_SHARED_INFO, Table,
660             Offset + SubOffset);

662         AcpiOsPrintf ("\n");
663         Status = AcpiDmDumpTable (Length, Offset + SubOffset, SharedInfoTable,
664                     sizeof (ACPI_CSRT_SHARED_INFO), AcpiDmTableInfoCsrt1);
665         if (ACPI_FAILURE (Status))
666         {
667             return;
668         }

670         SubOffset += SubTable->SharedInfoLength;

672         /* Sub-Subtables (Resource Descriptors) */

674         SubSubTable = ACPI_ADD_PTR (ACPI_CSRT_DESCRIPTOR, Table,
675             Offset + SubOffset);

677         while ((SubOffset < SubTable->Length) &&
678                ((Offset + SubOffset) < Table->Length))
679         {
680             AcpiOsPrintf ("\n");
681             Status = AcpiDmDumpTable (Length, Offset + SubOffset, SubSubTable,
682                         SubSubTable->Length, AcpiDmTableInfoCsrt2);
683             if (ACPI_FAILURE (Status))
684             {
685                 return;
686             }

688             SubSubOffset = sizeof (ACPI_CSRT_DESCRIPTOR);

690             /* Resource-specific info buffer */

692             InfoLength = SubSubTable->Length - SubSubOffset;

694             AcpiDmDumpBuffer (SubSubTable, SubSubOffset, InfoLength,
695                 Offset + SubOffset + SubSubOffset, "ResourceInfo");
696             SubSubOffset += InfoLength;

698             /* Point to next sub-subtable */

700             SubOffset += SubSubTable->Length;
701             SubSubTable = ACPI_ADD_PTR (ACPI_CSRT_DESCRIPTOR, SubSubTable,
702                         SubSubTable->Length);
703         }

705         /* Point to next sub-table */

707         Offset += SubTable->Length;
708         SubTable = ACPI_ADD_PTR (ACPI_CSRT_GROUP, SubTable,
709                     SubTable->Length);
710     }
711 }


714 /*******************************************************************************
715  *
716  * FUNCTION:    AcpiDmDumpDbg2
717  *
718  * PARAMETERS:  Table               - A DBG2 table
719  *
```



```
720  * RETURN:      None
721  *
722  * DESCRIPTION: Format the contents of a DBG2. This table type consists
723  *              of an open-ended number of subtables.
724  *
725  ******************************************************************************/

727 void
728 AcpiDmDumpDbg2 (
729     ACPI_TABLE_HEADER       *Table)
730 {
731     ACPI_STATUS             Status;
732     ACPI_DBG2_DEVICE        *SubTable;
733     UINT32                  Length = Table->Length;
734     UINT32                  Offset = sizeof (ACPI_TABLE_DBG2);
735     UINT32                  i;
736     UINT32                  ArrayOffset;
737     UINT32                  AbsoluteOffset;
738     UINT8                   *Array;


741     /* Main table */

743     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoDbg2);
744     if (ACPI_FAILURE (Status))
745     {
746         return;
747     }

749     /* Sub-tables */

751     SubTable = ACPI_ADD_PTR (ACPI_DBG2_DEVICE, Table, Offset);
752     while (Offset < Table->Length)
753     {
754         AcpiOsPrintf ("\n");
755         Status = AcpiDmDumpTable (Length, Offset, SubTable,
756                     SubTable->Length, AcpiDmTableInfoDbg2Device);
757         if (ACPI_FAILURE (Status))
758         {
759             return;
760         }

762         /* Dump the BaseAddress array */

764         for (i = 0; i < SubTable->RegisterCount; i++)
765         {
766             ArrayOffset = SubTable->BaseAddressOffset +
767                 (sizeof (ACPI_GENERIC_ADDRESS) * i);
768             AbsoluteOffset = Offset + ArrayOffset;
769             Array = (UINT8 *) SubTable + ArrayOffset;

771             Status = AcpiDmDumpTable (Length, AbsoluteOffset, Array,
772                         SubTable->Length, AcpiDmTableInfoDbg2Addr);
773             if (ACPI_FAILURE (Status))
774             {
775                 return;
776             }
777         }

779         /* Dump the AddressSize array */

781         for (i = 0; i < SubTable->RegisterCount; i++)
782         {
783             ArrayOffset = SubTable->AddressSizeOffset +
784                 (sizeof (UINT32) * i);
785             AbsoluteOffset = Offset + ArrayOffset;
```

```
786             Array = (UINT8 *) SubTable + ArrayOffset;

788             Status = AcpiDmDumpTable (Length, AbsoluteOffset, Array,
789                         SubTable->Length, AcpiDmTableInfoDbg2Size);
790             if (ACPI_FAILURE (Status))
791             {
792                 return;
793             }
794         }

796         /* Dump the Namestring (required) */

798         AcpiOsPrintf ("\n");
799         ArrayOffset = SubTable->NamepathOffset;
800         AbsoluteOffset = Offset + ArrayOffset;
801         Array = (UINT8 *) SubTable + ArrayOffset;

803         Status = AcpiDmDumpTable (Length, AbsoluteOffset, Array,
804                     SubTable->Length, AcpiDmTableInfoDbg2Name);
805         if (ACPI_FAILURE (Status))
806         {
807             return;
808         }

810         /* Dump the OemData (optional) */

812         if (SubTable->OemDataOffset)
813         {
814             AcpiDmDumpBuffer (SubTable, SubTable->OemDataOffset, SubTable->OemDa
815                 Offset + SubTable->OemDataOffset, "OEM Data");
816         }

818         /* Point to next sub-table */

820         Offset += SubTable->Length;
821         SubTable = ACPI_ADD_PTR (ACPI_DBG2_DEVICE, SubTable,
822                     SubTable->Length);
823     }
824 }


827 /*******************************************************************************
828  *
829  * FUNCTION:    AcpiDmDumpDmar
830  *
831  * PARAMETERS:  Table               - A DMAR table
832  *
833  * RETURN:      None
834  *
835  * DESCRIPTION: Format the contents of a DMAR. This table type consists
836  *              of an open-ended number of subtables.
837  *
838  ******************************************************************************/


841 void
842 AcpiDmDumpDmar (
843     ACPI_TABLE_HEADER       *Table)
844 {
845     ACPI_STATUS             Status;
846     ACPI_DMAR_HEADER        *SubTable;
847     UINT32                  Length = Table->Length;
848     UINT32                  Offset = sizeof (ACPI_TABLE_DMAR);
849     ACPI_DMTABLE_INFO       *InfoTable;
850     ACPI_DMAR_DEVICE_SCOPE  *ScopeTable;
851     UINT32                  ScopeOffset;
```



```
852     UINT8                   *PciPath;
853     UINT32                  PathOffset;


856     /* Main table */

858     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoDmar);
859     if (ACPI_FAILURE (Status))
860     {
861         return;
862     }

864     /* Sub-tables */

866     SubTable = ACPI_ADD_PTR (ACPI_DMAR_HEADER, Table, Offset);
867     while (Offset < Table->Length)
868     {
869         /* Common sub-table header */

871         AcpiOsPrintf ("\n");
872         Status = AcpiDmDumpTable (Length, Offset, SubTable,
873                     SubTable->Length, AcpiDmTableInfoDmarHdr);
874         if (ACPI_FAILURE (Status))
875         {
876             return;
877         }
878         AcpiOsPrintf ("\n");

880         switch (SubTable->Type)
881         {
882         case ACPI_DMAR_TYPE_HARDWARE_UNIT:

884             InfoTable = AcpiDmTableInfoDmar0;
885             ScopeOffset = sizeof (ACPI_DMAR_HARDWARE_UNIT);
886             break;

888         case ACPI_DMAR_TYPE_RESERVED_MEMORY:

890             InfoTable = AcpiDmTableInfoDmar1;
891             ScopeOffset = sizeof (ACPI_DMAR_RESERVED_MEMORY);
892             break;

894         case ACPI_DMAR_TYPE_ATSR:

896             InfoTable = AcpiDmTableInfoDmar2;
897             ScopeOffset = sizeof (ACPI_DMAR_ATSR);
898             break;

900         case ACPI_DMAR_HARDWARE_AFFINITY:

902             InfoTable = AcpiDmTableInfoDmar3;
903             ScopeOffset = sizeof (ACPI_DMAR_RHSA);
904             break;

906         default:

908             AcpiOsPrintf ("\n**** Unknown DMAR sub-table type 0x%X\n\n", SubTabl
909             return;
910         }

912         Status = AcpiDmDumpTable (Length, Offset, SubTable,
913                     SubTable->Length, InfoTable);
914         if (ACPI_FAILURE (Status))
915         {
916             return;
917         }
```

```
919         /* Dump the device scope entries (if any) */

921         ScopeTable = ACPI_ADD_PTR (ACPI_DMAR_DEVICE_SCOPE, SubTable, ScopeOffset
922         while (ScopeOffset < SubTable->Length)
923         {
924             AcpiOsPrintf ("\n");
925             Status = AcpiDmDumpTable (Length, Offset + ScopeOffset, ScopeTable,
926                         ScopeTable->Length, AcpiDmTableInfoDmarScope);
927             if (ACPI_FAILURE (Status))
928             {
929                 return;
930             }
931             AcpiOsPrintf ("\n");

933             /* Dump the PCI Path entries for this device scope */

935             PathOffset = sizeof (ACPI_DMAR_DEVICE_SCOPE); /* Path entries start

937             PciPath = ACPI_ADD_PTR (UINT8, ScopeTable,
938                 sizeof (ACPI_DMAR_DEVICE_SCOPE));

940             while (PathOffset < ScopeTable->Length)
941             {
942                 AcpiDmLineHeader ((PathOffset + ScopeOffset + Offset), 2, "PCI P
943                 AcpiOsPrintf ("%2.2X,%2.2X\n", PciPath[0], PciPath[1]);

945                 /* Point to next PCI Path entry */

947                 PathOffset += 2;
948                 PciPath += 2;
949                 AcpiOsPrintf ("\n");
950             }

952             /* Point to next device scope entry */

954             ScopeOffset += ScopeTable->Length;
955             ScopeTable = ACPI_ADD_PTR (ACPI_DMAR_DEVICE_SCOPE,
956                 ScopeTable, ScopeTable->Length);
957         }

959         /* Point to next sub-table */

961         Offset += SubTable->Length;
962         SubTable = ACPI_ADD_PTR (ACPI_DMAR_HEADER, SubTable, SubTable->Length);
963     }
964 }


967 /*******************************************************************************
968  *
969  * FUNCTION:    AcpiDmDumpEinj
970  *
971  * PARAMETERS:  Table               - A EINJ table
972  *
973  * RETURN:      None
974  *
975  * DESCRIPTION: Format the contents of a EINJ. This table type consists
976  *              of an open-ended number of subtables.
977  *
978  ******************************************************************************/

980 void
981 AcpiDmDumpEinj (
982     ACPI_TABLE_HEADER       *Table)
983 {
```



```
984     ACPI_STATUS             Status;
985     ACPI_WHEA_HEADER        *SubTable;
986     UINT32                  Length = Table->Length;
987     UINT32                  Offset = sizeof (ACPI_TABLE_EINJ);


990     /* Main table */

992     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoEinj);
993     if (ACPI_FAILURE (Status))
994     {
995         return;
996     }

998     /* Sub-tables */

1000    SubTable = ACPI_ADD_PTR (ACPI_WHEA_HEADER, Table, Offset);
1001    while (Offset < Table->Length)
1002    {
1003        AcpiOsPrintf ("\n");
1004        Status = AcpiDmDumpTable (Length, Offset, SubTable,
1005                    sizeof (ACPI_WHEA_HEADER), AcpiDmTableInfoEinj0);
1006        if (ACPI_FAILURE (Status))
1007        {
1008            return;
1009        }

1011        /* Point to next sub-table (each subtable is of fixed length) */

1013        Offset += sizeof (ACPI_WHEA_HEADER);
1014        SubTable = ACPI_ADD_PTR (ACPI_WHEA_HEADER, SubTable,
1015                        sizeof (ACPI_WHEA_HEADER));
1016    }
1017 }


1020 /*******************************************************************************
1021  *
1022  * FUNCTION:    AcpiDmDumpErst
1023  *
1024  * PARAMETERS:  Table               - A ERST table
1025  *
1026  * RETURN:      None
1027  *
1028  * DESCRIPTION: Format the contents of a ERST. This table type consists
1029  *              of an open-ended number of subtables.
1030  *
1031  ******************************************************************************/

1033 void
1034 AcpiDmDumpErst (
1035     ACPI_TABLE_HEADER       *Table)
1036 {
1037     ACPI_STATUS             Status;
1038     ACPI_WHEA_HEADER        *SubTable;
1039     UINT32                  Length = Table->Length;
1040     UINT32                  Offset = sizeof (ACPI_TABLE_ERST);


1043     /* Main table */

1045     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoErst);
1046     if (ACPI_FAILURE (Status))
1047     {
1048         return;
1049     }
```

```
1051     /* Sub-tables */

1053     SubTable = ACPI_ADD_PTR (ACPI_WHEA_HEADER, Table, Offset);
1054     while (Offset < Table->Length)
1055     {
1056         AcpiOsPrintf ("\n");
1057         Status = AcpiDmDumpTable (Length, Offset, SubTable,
1058                     sizeof (ACPI_WHEA_HEADER), AcpiDmTableInfoErst0);
1059         if (ACPI_FAILURE (Status))
1060         {
1061             return;
1062         }

1064         /* Point to next sub-table (each subtable is of fixed length) */

1066         Offset += sizeof (ACPI_WHEA_HEADER);
1067         SubTable = ACPI_ADD_PTR (ACPI_WHEA_HEADER, SubTable,
1068                         sizeof (ACPI_WHEA_HEADER));
1069     }
1070 }


1073 /******************************************************************************
1074  *
1075  * FUNCTION:    AcpiDmDumpFpdt
1076  *
1077  * PARAMETERS:  Table               - A FPDT table
1078  *
1079  * RETURN:      None
1080  *
1081  * DESCRIPTION: Format the contents of a FPDT. This table type consists
1082  *              of an open-ended number of subtables.
1083  *
1084  *****************************************************************************/

1086 void
1087 AcpiDmDumpFpdt (
1088     ACPI_TABLE_HEADER       *Table)
1089 {
1090     ACPI_STATUS             Status;
1091     ACPI_FPDT_HEADER        *SubTable;
1092     UINT32                  Length = Table->Length;
1093     UINT32                  Offset = sizeof (ACPI_TABLE_FPDT);
1094     ACPI_DMTABLE_INFO       *InfoTable;


1097     /* There is no main table (other than the standard ACPI header) */

1099     /* Sub-tables */

1101     SubTable = ACPI_ADD_PTR (ACPI_FPDT_HEADER, Table, Offset);
1102     while (Offset < Table->Length)
1103     {
1104         /* Common sub-table header */

1106         AcpiOsPrintf ("\n");
1107         Status = AcpiDmDumpTable (Length, Offset, SubTable,
1108                     SubTable->Length, AcpiDmTableInfoFpdtHdr);
1109         if (ACPI_FAILURE (Status))
1110         {
1111             return;
1112         }

1114         switch (SubTable->Type)
1115         {
```



```
1116         case ACPI_FPDT_TYPE_BOOT:

1118             InfoTable = AcpiDmTableInfoFpdt0;
1119             break;

1121         case ACPI_FPDT_TYPE_S3PERF:

1123             InfoTable = AcpiDmTableInfoFpdt1;
1124             break;

1126         default:

1128             AcpiOsPrintf ("\n**** Unknown FPDT sub-table type 0x%X\n\n", SubTabl

1130             /* Attempt to continue */

1132             if (!SubTable->Length)
1133             {
1134                 AcpiOsPrintf ("Invalid zero length subtable\n");
1135                 return;
1136             }
1137             goto NextSubTable;
1138         }

1140         Status = AcpiDmDumpTable (Length, Offset, SubTable,
1141                     SubTable->Length, InfoTable);
1142         if (ACPI_FAILURE (Status))
1143         {
1144             return;
1145         }

1147 NextSubTable:
1148         /* Point to next sub-table */

1150         Offset += SubTable->Length;
1151         SubTable = ACPI_ADD_PTR (ACPI_FPDT_HEADER, SubTable, SubTable->Length);
1152     }
1153 }


1156 /******************************************************************************
1157  *
1158  * FUNCTION:    AcpiDmDumpHest
1159  *
1160  * PARAMETERS:  Table               - A HEST table
1161  *
1162  * RETURN:      None
1163  *
1164  * DESCRIPTION: Format the contents of a HEST. This table type consists
1165  *              of an open-ended number of subtables.
1166  *
1167  *****************************************************************************/

1169 void
1170 AcpiDmDumpHest (
1171     ACPI_TABLE_HEADER       *Table)
1172 {
1173     ACPI_STATUS             Status;
1174     ACPI_HEST_HEADER        *SubTable;
1175     UINT32                  Length = Table->Length;
1176     UINT32                  Offset = sizeof (ACPI_TABLE_HEST);
1177     ACPI_DMTABLE_INFO       *InfoTable;
1178     UINT32                  SubTableLength;
1179     UINT32                  BankCount;
1180     ACPI_HEST_IA_ERROR_BANK *BankTable;
```

```
1183     /* Main table */
1185     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoHest);
1186     if (ACPI_FAILURE (Status))
1187     {
1188         return;
1189     }
1191     /* Sub-tables */
1193     SubTable = ACPI_ADD_PTR (ACPI_HEST_HEADER, Table, Offset);
1194     while (Offset < Table->Length)
1195     {
1196         BankCount = 0;
1197         switch (SubTable->Type)
1198         {
1199         case ACPI_HEST_TYPE_IA32_CHECK:
1201             InfoTable = AcpiDmTableInfoHest0;
1202             SubTableLength = sizeof (ACPI_HEST_IA_MACHINE_CHECK);
1203             BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_MACHINE_CHECK,
1204                         SubTable))->NumHardwareBanks;
1205             break;
1207         case ACPI_HEST_TYPE_IA32_CORRECTED_CHECK:
1209             InfoTable = AcpiDmTableInfoHest1;
1210             SubTableLength = sizeof (ACPI_HEST_IA_CORRECTED);
1211             BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_CORRECTED,
1212                         SubTable))->NumHardwareBanks;
1213             break;
1215         case ACPI_HEST_TYPE_IA32_NMI:
1217             InfoTable = AcpiDmTableInfoHest2;
1218             SubTableLength = sizeof (ACPI_HEST_IA_NMI);
1219             break;
1221         case ACPI_HEST_TYPE_AER_ROOT_PORT:
1223             InfoTable = AcpiDmTableInfoHest6;
1224             SubTableLength = sizeof (ACPI_HEST_AER_ROOT);
1225             break;
1227         case ACPI_HEST_TYPE_AER_ENDPOINT:
1229             InfoTable = AcpiDmTableInfoHest7;
1230             SubTableLength = sizeof (ACPI_HEST_AER);
1231             break;
1233         case ACPI_HEST_TYPE_AER_BRIDGE:
1235             InfoTable = AcpiDmTableInfoHest8;
1236             SubTableLength = sizeof (ACPI_HEST_AER_BRIDGE);
1237             break;
1239         case ACPI_HEST_TYPE_GENERIC_ERROR:
1241             InfoTable = AcpiDmTableInfoHest9;
1242             SubTableLength = sizeof (ACPI_HEST_GENERIC);
1243             break;
1245         default:
1247             /* Cannot continue on unknown type - no length */
```



```
1249             AcpiOsPrintf ("\n**** Unknown HEST sub-table type 0x%X\n", SubTable-
1250             return;
1251         }
1253         AcpiOsPrintf ("\n");
1254         Status = AcpiDmDumpTable (Length, Offset, SubTable,
1255                     SubTableLength, InfoTable);
1256         if (ACPI_FAILURE (Status))
1257         {
1258             return;
1259         }
1261         /* Point to end of current subtable (each subtable above is of fixed len
1263         Offset += SubTableLength;
1265         /* If there are any (fixed-length) Error Banks from above, dump them now
1267         if (BankCount)
1268         {
1269             BankTable = ACPI_ADD_PTR (ACPI_HEST_IA_ERROR_BANK, SubTable, SubTabl
1270             SubTableLength += BankCount * sizeof (ACPI_HEST_IA_ERROR_BANK);
1272             while (BankCount)
1273             {
1274                 AcpiOsPrintf ("\n");
1275                 Status = AcpiDmDumpTable (Length, Offset, BankTable,
1276                             sizeof (ACPI_HEST_IA_ERROR_BANK), AcpiDmTableInfoHes
1277                 if (ACPI_FAILURE (Status))
1278                 {
1279                     return;
1280                 }
1281                 Offset += sizeof (ACPI_HEST_IA_ERROR_BANK);
1282                 BankTable++;
1283                 BankCount--;
1284             }
1285         }
1287         /* Point to next sub-table */
1289         SubTable = ACPI_ADD_PTR (ACPI_HEST_HEADER, SubTable, SubTableLength);
1290     }
1291 }


1294 /*******************************************************************************
1295  *
1296  * FUNCTION:    AcpiDmDumpIvrs
1297  *
1298  * PARAMETERS:  Table               - A IVRS table
1299  *
1300  * RETURN:      None
1301  *
1302  * DESCRIPTION: Format the contents of a IVRS
1303  *
1304  ******************************************************************************/

1306 static UINT8 EntrySizes[] = {4,8,16,32};

1308 void
1309 AcpiDmDumpIvrs (
1310     ACPI_TABLE_HEADER       *Table)
1311 {
1312     ACPI_STATUS             Status;
1313     UINT32                  Offset = sizeof (ACPI_TABLE_IVRS);
```

```
1314    UINT32                  EntryOffset;
1315    UINT32                  EntryLength;
1316    UINT32                  EntryType;
1317    ACPI_IVRS_DE_HEADER     *DeviceEntry;
1318    ACPI_IVRS_HEADER        *SubTable;
1319    ACPI_DMTABLE_INFO       *InfoTable;


1322    /* Main table */

1324    Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoIvrs);
1325    if (ACPI_FAILURE (Status))
1326    {
1327        return;
1328    }

1330    /* Sub-tables */

1332    SubTable = ACPI_ADD_PTR (ACPI_IVRS_HEADER, Table, Offset);
1333    while (Offset < Table->Length)
1334    {
1335        /* Common sub-table header */

1337        AcpiOsPrintf ("\n");
1338        Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
1339                    SubTable->Length, AcpiDmTableInfoIvrsHdr);
1340        if (ACPI_FAILURE (Status))
1341        {
1342            return;
1343        }

1345        switch (SubTable->Type)
1346        {
1347        case ACPI_IVRS_TYPE_HARDWARE:

1349            InfoTable = AcpiDmTableInfoIvrs0;
1350            break;

1352        case ACPI_IVRS_TYPE_MEMORY1:
1353        case ACPI_IVRS_TYPE_MEMORY2:
1354        case ACPI_IVRS_TYPE_MEMORY3:

1356            InfoTable = AcpiDmTableInfoIvrs1;
1357            break;

1359        default:

1361            AcpiOsPrintf ("\n**** Unknown IVRS sub-table type 0x%X\n",
1362                SubTable->Type);

1364            /* Attempt to continue */

1366            if (!SubTable->Length)
1367            {
1368                AcpiOsPrintf ("Invalid zero length subtable\n");
1369                return;
1370            }
1371            goto NextSubTable;
1372        }

1374        /* Dump the subtable */

1376        AcpiOsPrintf ("\n");
1377        Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
1378                    SubTable->Length, InfoTable);
1379        if (ACPI_FAILURE (Status))
```



```
1380        {
1381            return;
1382        }

1384        /* The hardware subtable can contain multiple device entries */

1386        if (SubTable->Type == ACPI_IVRS_TYPE_HARDWARE)
1387        {
1388            EntryOffset = Offset + sizeof (ACPI_IVRS_HARDWARE);
1389            DeviceEntry = ACPI_ADD_PTR (ACPI_IVRS_DE_HEADER, SubTable,
1390                            sizeof (ACPI_IVRS_HARDWARE));

1392            while (EntryOffset < (Offset + SubTable->Length))
1393            {
1394                AcpiOsPrintf ("\n");
1395                /*
1396                 * Upper 2 bits of Type encode the length of the device entry
1397                 *
1398                 * 00 = 4 byte
1399                 * 01 = 8 byte
1400                 * 10 = 16 byte - currently no entries defined
1401                 * 11 = 32 byte - currently no entries defined
1402                 */
1403                EntryType = DeviceEntry->Type;
1404                EntryLength = EntrySizes [EntryType >> 6];

1406                switch (EntryType)
1407                {
1408                /* 4-byte device entries */

1410                case ACPI_IVRS_TYPE_PAD4:
1411                case ACPI_IVRS_TYPE_ALL:
1412                case ACPI_IVRS_TYPE_SELECT:
1413                case ACPI_IVRS_TYPE_START:
1414                case ACPI_IVRS_TYPE_END:

1416                    InfoTable = AcpiDmTableInfoIvrs4;
1417                    break;

1419                /* 8-byte entries, type A */

1421                case ACPI_IVRS_TYPE_ALIAS_SELECT:
1422                case ACPI_IVRS_TYPE_ALIAS_START:

1424                    InfoTable = AcpiDmTableInfoIvrs8a;
1425                    break;

1427                /* 8-byte entries, type B */

1429                case ACPI_IVRS_TYPE_PAD8:
1430                case ACPI_IVRS_TYPE_EXT_SELECT:
1431                case ACPI_IVRS_TYPE_EXT_START:

1433                    InfoTable = AcpiDmTableInfoIvrs8b;
1434                    break;

1436                /* 8-byte entries, type C */

1438                case ACPI_IVRS_TYPE_SPECIAL:

1440                    InfoTable = AcpiDmTableInfoIvrs8c;
1441                    break;

1443                default:
1444                    InfoTable = AcpiDmTableInfoIvrs4;
1445                    AcpiOsPrintf (
```

```
                        "\n**** Unknown IVRS device entry type/length: "
                        "0x%.2X/0x%X at offset 0x%.4X: (header below)\n",
                        EntryType, EntryLength, EntryOffset);
                    break;
                }

                /* Dump the Device Entry */

                Status = AcpiDmDumpTable (Table->Length, EntryOffset,
                            DeviceEntry, EntryLength, InfoTable);

                EntryOffset += EntryLength;
                DeviceEntry = ACPI_ADD_PTR (ACPI_IVRS_DE_HEADER, DeviceEntry,
                                EntryLength);
            }
        }

NextSubTable:
        /* Point to next sub-table */

        Offset += SubTable->Length;
        SubTable = ACPI_ADD_PTR (ACPI_IVRS_HEADER, SubTable, SubTable->Length);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpMadt
 *
 * PARAMETERS:  Table               - A MADT table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a MADT. This table type consists
 *              of an open-ended number of subtables.
 *
 ******************************************************************************/

void
AcpiDmDumpMadt (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    ACPI_SUBTABLE_HEADER    *SubTable;
    UINT32                  Length = Table->Length;
    UINT32                  Offset = sizeof (ACPI_TABLE_MADT);
    ACPI_DMTABLE_INFO       *InfoTable;


    /* Main table */

    Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoMadt);
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Sub-tables */

    SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, Table, Offset);
    while (Offset < Table->Length)
    {
        /* Common sub-table header */

        AcpiOsPrintf ("\n");
```



```
        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Length, AcpiDmTableInfoMadtHdr);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        switch (SubTable->Type)
        {
        case ACPI_MADT_TYPE_LOCAL_APIC:

            InfoTable = AcpiDmTableInfoMadt0;
            break;

        case ACPI_MADT_TYPE_IO_APIC:

            InfoTable = AcpiDmTableInfoMadt1;
            break;

        case ACPI_MADT_TYPE_INTERRUPT_OVERRIDE:

            InfoTable = AcpiDmTableInfoMadt2;
            break;

        case ACPI_MADT_TYPE_NMI_SOURCE:

            InfoTable = AcpiDmTableInfoMadt3;
            break;

        case ACPI_MADT_TYPE_LOCAL_APIC_NMI:

            InfoTable = AcpiDmTableInfoMadt4;
            break;

        case ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE:

            InfoTable = AcpiDmTableInfoMadt5;
            break;

        case ACPI_MADT_TYPE_IO_SAPIC:

            InfoTable = AcpiDmTableInfoMadt6;
            break;

        case ACPI_MADT_TYPE_LOCAL_SAPIC:

            InfoTable = AcpiDmTableInfoMadt7;
            break;

        case ACPI_MADT_TYPE_INTERRUPT_SOURCE:

            InfoTable = AcpiDmTableInfoMadt8;
            break;

        case ACPI_MADT_TYPE_LOCAL_X2APIC:

            InfoTable = AcpiDmTableInfoMadt9;
            break;

        case ACPI_MADT_TYPE_LOCAL_X2APIC_NMI:

            InfoTable = AcpiDmTableInfoMadt10;
            break;

        case ACPI_MADT_TYPE_GENERIC_INTERRUPT:
```

```
            InfoTable = AcpiDmTableInfoMadt11;
            break;

        case ACPI_MADT_TYPE_GENERIC_DISTRIBUTOR:

            InfoTable = AcpiDmTableInfoMadt12;
            break;

        default:

            AcpiOsPrintf ("\n**** Unknown MADT sub-table type 0x%X\n\n", SubTabl

            /* Attempt to continue */

            if (!SubTable->Length)
            {
                AcpiOsPrintf ("Invalid zero length subtable\n");
                return;
            }
            goto NextSubTable;
        }

        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Length, InfoTable);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

NextSubTable:
        /* Point to next sub-table */

        Offset += SubTable->Length;
        SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, SubTable, SubTable->Lengt
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpMcfg
 *
 * PARAMETERS:  Table               - A MCFG Table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a MCFG table
 *
 ******************************************************************************/

void
AcpiDmDumpMcfg (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    UINT32                  Offset = sizeof (ACPI_TABLE_MCFG);
    ACPI_MCFG_ALLOCATION    *SubTable;


    /* Main table */

    Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMcfg);
    if (ACPI_FAILURE (Status))
    {
        return;
    }
```



```
    /* Sub-tables */

    SubTable = ACPI_ADD_PTR (ACPI_MCFG_ALLOCATION, Table, Offset);
    while (Offset < Table->Length)
    {
        if (Offset + sizeof (ACPI_MCFG_ALLOCATION) > Table->Length)
        {
            AcpiOsPrintf ("Warning: there are %u invalid trailing bytes\n",
                sizeof (ACPI_MCFG_ALLOCATION) - (Offset - Table->Length));
            return;
        }

        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
                    sizeof (ACPI_MCFG_ALLOCATION), AcpiDmTableInfoMcfg0);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Point to next sub-table (each subtable is of fixed length) */

        Offset += sizeof (ACPI_MCFG_ALLOCATION);
        SubTable = ACPI_ADD_PTR (ACPI_MCFG_ALLOCATION, SubTable,
                        sizeof (ACPI_MCFG_ALLOCATION));
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpMpst
 *
 * PARAMETERS:  Table               - A MPST Table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a MPST table
 *
 ******************************************************************************/

void
AcpiDmDumpMpst (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    UINT32                  Offset = sizeof (ACPI_TABLE_MPST);
    ACPI_MPST_POWER_NODE    *SubTable0;
    ACPI_MPST_POWER_STATE   *SubTable0A;
    ACPI_MPST_COMPONENT     *SubTable0B;
    ACPI_MPST_DATA_HDR      *SubTable1;
    ACPI_MPST_POWER_DATA    *SubTable2;
    UINT16                  SubtableCount;
    UINT32                  PowerStateCount;
    UINT32                  ComponentCount;


    /* Main table */

    Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMpst);
    if (ACPI_FAILURE (Status))
    {
        return;
    }
```

```
1710     /* Subtable: Memory Power Node(s) */

1712     SubtableCount = (ACPI_CAST_PTR (ACPI_TABLE_MPST, Table))->PowerNodeCount;
1713     SubTable0 = ACPI_ADD_PTR (ACPI_MPST_POWER_NODE, Table, Offset);

1715     while ((Offset < Table->Length) && SubtableCount)
1716     {
1717         AcpiOsPrintf ("\n");
1718         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable0,
1719                     sizeof (ACPI_MPST_POWER_NODE), AcpiDmTableInfoMpst0);
1720         if (ACPI_FAILURE (Status))
1721         {
1722             return;
1723         }

1725         /* Extract the sub-subtable counts */

1727         PowerStateCount = SubTable0->NumPowerStates;
1728         ComponentCount = SubTable0->NumPhysicalComponents;
1729         Offset += sizeof (ACPI_MPST_POWER_NODE);

1731         /* Sub-subtables - Memory Power State Structure(s) */

1733         SubTable0A = ACPI_ADD_PTR (ACPI_MPST_POWER_STATE, SubTable0,
1734             sizeof (ACPI_MPST_POWER_NODE));

1736         while (PowerStateCount)
1737         {
1738             AcpiOsPrintf ("\n");
1739             Status = AcpiDmDumpTable (Table->Length, Offset, SubTable0A,
1740                         sizeof (ACPI_MPST_POWER_STATE), AcpiDmTableInfoMpst0A);
1741             if (ACPI_FAILURE (Status))
1742             {
1743                 return;
1744             }

1746             SubTable0A++;
1747             PowerStateCount--;
1748             Offset += sizeof (ACPI_MPST_POWER_STATE);
1749         }

1751         /* Sub-subtables - Physical Component ID Structure(s) */

1753         SubTable0B = ACPI_CAST_PTR (ACPI_MPST_COMPONENT, SubTable0A);

1755         if (ComponentCount)
1756         {
1757             AcpiOsPrintf ("\n");
1758         }

1760         while (ComponentCount)
1761         {
1762             Status = AcpiDmDumpTable (Table->Length, Offset, SubTable0B,
1763                         sizeof (ACPI_MPST_COMPONENT), AcpiDmTableInfoMpst0B);
1764             if (ACPI_FAILURE (Status))
1765             {
1766                 return;
1767             }

1769             SubTable0B++;
1770             ComponentCount--;
1771             Offset += sizeof (ACPI_MPST_COMPONENT);
1772         }

1774         /* Point to next Memory Power Node subtable */
```



```
1776         SubtableCount--;
1777         SubTable0 = ACPI_ADD_PTR (ACPI_MPST_POWER_NODE, SubTable0,
1778             sizeof (ACPI_MPST_POWER_NODE) +
1779             (sizeof (ACPI_MPST_POWER_STATE) * SubTable0->NumPowerStates) +
1780             (sizeof (ACPI_MPST_COMPONENT) * SubTable0->NumPhysicalComponents));
1781     }

1783     /* Subtable: Count of Memory Power State Characteristic structures */

1785     AcpiOsPrintf ("\n");
1786     SubTable1 = ACPI_CAST_PTR (ACPI_MPST_DATA_HDR, SubTable0);
1787     Status = AcpiDmDumpTable (Table->Length, Offset, SubTable1,
1788                 sizeof (ACPI_MPST_DATA_HDR), AcpiDmTableInfoMpst1);
1789     if (ACPI_FAILURE (Status))
1790     {
1791         return;
1792     }

1794     SubtableCount = SubTable1->CharacteristicsCount;
1795     Offset += sizeof (ACPI_MPST_DATA_HDR);

1797     /* Subtable: Memory Power State Characteristics structure(s) */

1799     SubTable2 = ACPI_ADD_PTR (ACPI_MPST_POWER_DATA, SubTable1, sizeof (ACPI_MPST

1801     while ((Offset < Table->Length) && SubtableCount)
1802     {
1803         AcpiOsPrintf ("\n");
1804         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable2,
1805                     sizeof (ACPI_MPST_POWER_DATA), AcpiDmTableInfoMpst2);
1806         if (ACPI_FAILURE (Status))
1807         {
1808             return;
1809         }

1811         SubTable2++;
1812         SubtableCount--;
1813         Offset += sizeof (ACPI_MPST_POWER_DATA);
1814     }
1815 }


1818 /*******************************************************************************
1819  *
1820  * FUNCTION:    AcpiDmDumpMsct
1821  *
1822  * PARAMETERS:  Table               - A MSCT table
1823  *
1824  * RETURN:      None
1825  *
1826  * DESCRIPTION: Format the contents of a MSCT
1827  *
1828  ******************************************************************************/

1830 void
1831 AcpiDmDumpMsct (
1832     ACPI_TABLE_HEADER       *Table)
1833 {
1834     ACPI_STATUS             Status;
1835     UINT32                  Offset = sizeof (ACPI_TABLE_MSCT);
1836     ACPI_MSCT_PROXIMITY     *SubTable;


1839     /* Main table */

1841     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMsct);
```

```
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Sub-tables */

    SubTable = ACPI_ADD_PTR (ACPI_MSCT_PROXIMITY, Table, Offset);
    while (Offset < Table->Length)
    {
        /* Common sub-table header */

        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
                    sizeof (ACPI_MSCT_PROXIMITY), AcpiDmTableInfoMsct0);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Point to next sub-table */

        Offset += sizeof (ACPI_MSCT_PROXIMITY);
        SubTable = ACPI_ADD_PTR (ACPI_MSCT_PROXIMITY, SubTable, sizeof (ACPI_MSC
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpMtmr
 *
 * PARAMETERS:  Table               - A MTMR table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a MTMR
 *
 ******************************************************************************/

void
AcpiDmDumpMtmr (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    UINT32                  Offset = sizeof (ACPI_TABLE_MTMR);
    ACPI_MTMR_ENTRY         *SubTable;


    /* Main table */

    Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMtmr);
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Sub-tables */

    SubTable = ACPI_ADD_PTR (ACPI_MTMR_ENTRY, Table, Offset);
    while (Offset < Table->Length)
    {
        /* Common sub-table header */

        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
```



```
                    sizeof (ACPI_MTMR_ENTRY), AcpiDmTableInfoMtmr0);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Point to next sub-table */

        Offset += sizeof (ACPI_MTMR_ENTRY);
        SubTable = ACPI_ADD_PTR (ACPI_MTMR_ENTRY, SubTable, sizeof (ACPI_MTMR_EN
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpPcct
 *
 * PARAMETERS:  Table               - A PCCT table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a PCCT. This table type consists
 *              of an open-ended number of subtables.
 *
 ******************************************************************************/

void
AcpiDmDumpPcct (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    ACPI_PCCT_SUBSPACE      *SubTable;
    UINT32                  Length = Table->Length;
    UINT32                  Offset = sizeof (ACPI_TABLE_PCCT);


    /* Main table */

    Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoPcct);
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Subtables */

    SubTable = ACPI_ADD_PTR (ACPI_PCCT_SUBSPACE, Table, Offset);
    while (Offset < Table->Length)
    {
        /* Common subtable header */

        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Header.Length, AcpiDmTableInfoPcctHdr);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* ACPI 5.0: Only one type of PCCT subtable is supported */

        if (SubTable->Header.Type != ACPI_PCCT_TYPE_GENERIC_SUBSPACE)
        {
            AcpiOsPrintf (
                "\n**** Unexpected or unknown PCCT subtable type 0x%X\n\n",
```

```
                SubTable->Header.Type);
            return;
        }

        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Header.Length, AcpiDmTableInfoPcct0);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Point to next subtable */

        Offset += SubTable->Header.Length;
        SubTable = ACPI_ADD_PTR (ACPI_PCCT_SUBSPACE, SubTable,
                    SubTable->Header.Length);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpPmtt
 *
 * PARAMETERS:  Table               - A PMTT table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Format the contents of a PMTT. This table type consists
 *              of an open-ended number of subtables.
 *
 ******************************************************************************/

void
AcpiDmDumpPmtt (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    ACPI_PMTT_HEADER        *SubTable;
    ACPI_PMTT_HEADER        *MemSubTable;
    ACPI_PMTT_HEADER        *DimmSubTable;
    ACPI_PMTT_DOMAIN        *DomainArray;
    UINT32                  Length = Table->Length;
    UINT32                  Offset = sizeof (ACPI_TABLE_PMTT);
    UINT32                  MemOffset;
    UINT32                  DimmOffset;
    UINT32                  DomainOffset;
    UINT32                  DomainCount;


    /* Main table */

    Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoPmtt);
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Subtables */

    SubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER, Table, Offset);
    while (Offset < Table->Length)
    {
        /* Common subtable header */
```



```
        AcpiOsPrintf ("\n");
        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Length, AcpiDmTableInfoPmttHdr);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Only Socket subtables are expected at this level */

        if (SubTable->Type != ACPI_PMTT_TYPE_SOCKET)
        {
            AcpiOsPrintf (
                "\n**** Unexpected or unknown PMTT subtable type 0x%X\n\n",
                SubTable->Type);
            return;
        }

        /* Dump the fixed-length portion of the subtable */

        Status = AcpiDmDumpTable (Length, Offset, SubTable,
                    SubTable->Length, AcpiDmTableInfoPmtt0);
        if (ACPI_FAILURE (Status))
        {
            return;
        }

        /* Walk the memory controller subtables */

        MemOffset = sizeof (ACPI_PMTT_SOCKET);
        MemSubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER, SubTable,
            sizeof (ACPI_PMTT_SOCKET));

        while (((Offset + MemOffset) < Table->Length) &&
            (MemOffset < SubTable->Length))
        {
            /* Common subtable header */

            AcpiOsPrintf ("\n");
            Status = AcpiDmDumpTable (Length,
                        Offset + MemOffset, MemSubTable,
                        MemSubTable->Length, AcpiDmTableInfoPmttHdr);
            if (ACPI_FAILURE (Status))
            {
                return;
            }

            /* Only memory controller subtables are expected at this level */

            if (MemSubTable->Type != ACPI_PMTT_TYPE_CONTROLLER)
            {
                AcpiOsPrintf (
                    "\n**** Unexpected or unknown PMTT subtable type 0x%X\n\n",
                    MemSubTable->Type);
                return;
            }

            /* Dump the fixed-length portion of the controller subtable */

            Status = AcpiDmDumpTable (Length,
                        Offset + MemOffset, MemSubTable,
                        MemSubTable->Length, AcpiDmTableInfoPmtt1);
            if (ACPI_FAILURE (Status))
            {
                return;
            }
```

```
2107            /* Walk the variable count of proximity domains */

2109            DomainCount = ((ACPI_PMTT_CONTROLLER *) MemSubTable)->DomainCount;
2110            DomainOffset = sizeof (ACPI_PMTT_CONTROLLER);
2111            DomainArray = ACPI_ADD_PTR (ACPI_PMTT_DOMAIN, MemSubTable,
2112                sizeof (ACPI_PMTT_CONTROLLER));

2114            while (((Offset + MemOffset + DomainOffset) < Table->Length) &&
2115                ((MemOffset + DomainOffset) < SubTable->Length) &&
2116                DomainCount)
2117            {
2118                Status = AcpiDmDumpTable (Length,
2119                            Offset + MemOffset + DomainOffset, DomainArray,
2120                            sizeof (ACPI_PMTT_DOMAIN), AcpiDmTableInfoPmtt1a);
2121                if (ACPI_FAILURE (Status))
2122                {
2123                    return;
2124                }

2126                DomainOffset += sizeof (ACPI_PMTT_DOMAIN);
2127                DomainArray++;
2128                DomainCount--;
2129            }

2131            if (DomainCount)
2132            {
2133                AcpiOsPrintf (
2134                    "\n**** DomainCount exceeds subtable length\n\n",
2135                    MemSubTable->Type);
2136            }

2138            /* Walk the physical component (DIMM) subtables */

2140            DimmOffset = DomainOffset;
2141            DimmSubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER, MemSubTable,
2142                DomainOffset);

2144            while (((Offset + MemOffset + DimmOffset) < Table->Length) &&
2145                (DimmOffset < MemSubTable->Length))
2146            {
2147                /* Common subtable header */

2149                AcpiOsPrintf ("\n");
2150                Status = AcpiDmDumpTable (Length,
2151                            Offset + MemOffset + DimmOffset, DimmSubTable,
2152                            DimmSubTable->Length, AcpiDmTableInfoPmttHdr);
2153                if (ACPI_FAILURE (Status))
2154                {
2155                    return;
2156                }

2158                /* Only DIMM subtables are expected at this level */

2160                if (DimmSubTable->Type != ACPI_PMTT_TYPE_DIMM)
2161                {
2162                    AcpiOsPrintf (
2163                        "\n**** Unexpected or unknown PMTT subtable type 0x%X\n\
2164                        DimmSubTable->Type);
2165                    return;
2166                }

2168                /* Dump the fixed-length DIMM subtable */

2170                Status = AcpiDmDumpTable (Length,
2171                            Offset + MemOffset + DimmOffset, DimmSubTable,
```



```
2172                            DimmSubTable->Length, AcpiDmTableInfoPmtt2);
2173                if (ACPI_FAILURE (Status))
2174                {
2175                    return;
2176                }

2178                /* Point to next DIMM subtable */

2180                DimmOffset += DimmSubTable->Length;
2181                DimmSubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER,
2182                    DimmSubTable, DimmSubTable->Length);
2183            }

2185            /* Point to next Controller subtable */

2187            MemOffset += MemSubTable->Length;
2188            MemSubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER,
2189                MemSubTable, MemSubTable->Length);
2190        }

2192        /* Point to next Socket subtable */

2194        Offset += SubTable->Length;
2195        SubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER,
2196            SubTable, SubTable->Length);
2197    }
2198 }


2201 /*******************************************************************************
2202  *
2203  * FUNCTION:    AcpiDmDumpS3pt
2204  *
2205  * PARAMETERS:  Table               - A S3PT table
2206  *
2207  * RETURN:      Length of the table
2208  *
2209  * DESCRIPTION: Format the contents of a S3PT
2210  *
2211  ******************************************************************************/

2213 UINT32
2214 AcpiDmDumpS3pt (
2215     ACPI_TABLE_HEADER       *Tables)
2216 {
2217     ACPI_STATUS             Status;
2218     UINT32                  Offset = sizeof (ACPI_TABLE_S3PT);
2219     ACPI_S3PT_HEADER        *SubTable;
2220     ACPI_DMTABLE_INFO       *InfoTable;
2221     ACPI_TABLE_S3PT         *S3ptTable = ACPI_CAST_PTR (ACPI_TABLE_S3PT, Tables)


2224     /* Main table */

2226     Status = AcpiDmDumpTable (Offset, 0, S3ptTable, 0, AcpiDmTableInfoS3pt);
2227     if (ACPI_FAILURE (Status))
2228     {
2229         return 0;
2230     }

2232     SubTable = ACPI_ADD_PTR (ACPI_S3PT_HEADER, S3ptTable, Offset);
2233     while (Offset < S3ptTable->Length)
2234     {
2235         /* Common sub-table header */

2237         AcpiOsPrintf ("\n");
```

```
2238         Status = AcpiDmDumpTable (S3ptTable->Length, Offset, SubTable,
2239                     SubTable->Length, AcpiDmTableInfoS3ptHdr);
2240         if (ACPI_FAILURE (Status))
2241         {
2242             return 0;
2243         }

2245         switch (SubTable->Type)
2246         {
2247         case ACPI_S3PT_TYPE_RESUME:

2249             InfoTable = AcpiDmTableInfoS3pt0;
2250             break;

2252         case ACPI_S3PT_TYPE_SUSPEND:

2254             InfoTable = AcpiDmTableInfoS3pt1;
2255             break;

2257         default:

2259             AcpiOsPrintf ("\n**** Unknown S3PT sub-table type 0x%X\n", SubTable-

2261             /* Attempt to continue */

2263             if (!SubTable->Length)
2264             {
2265                 AcpiOsPrintf ("Invalid zero length subtable\n");
2266                 return 0;
2267             }
2268             goto NextSubTable;
2269         }

2271         AcpiOsPrintf ("\n");
2272         Status = AcpiDmDumpTable (S3ptTable->Length, Offset, SubTable,
2273                     SubTable->Length, InfoTable);
2274         if (ACPI_FAILURE (Status))
2275         {
2276             return 0;
2277         }

2279 NextSubTable:
2280         /* Point to next sub-table */

2282         Offset += SubTable->Length;
2283         SubTable = ACPI_ADD_PTR (ACPI_S3PT_HEADER, SubTable, SubTable->Length);
2284     }

2286     return (S3ptTable->Length);
2287 }


2290 /*******************************************************************************
2291  *
2292  * FUNCTION:    AcpiDmDumpSlic
2293  *
2294  * PARAMETERS:  Table               - A SLIC table
2295  *
2296  * RETURN:      None
2297  *
2298  * DESCRIPTION: Format the contents of a SLIC
2299  *
2300  ******************************************************************************/

2302 void
2303 AcpiDmDumpSlic (
```



```
2304     ACPI_TABLE_HEADER       *Table)
2305 {
2306     ACPI_STATUS             Status;
2307     UINT32                  Offset = sizeof (ACPI_TABLE_SLIC);
2308     ACPI_SLIC_HEADER        *SubTable;
2309     ACPI_DMTABLE_INFO       *InfoTable;


2312     /* There is no main SLIC table, only subtables */

2314     SubTable = ACPI_ADD_PTR (ACPI_SLIC_HEADER, Table, Offset);
2315     while (Offset < Table->Length)
2316     {
2317         /* Common sub-table header */

2319         AcpiOsPrintf ("\n");
2320         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2321                     SubTable->Length, AcpiDmTableInfoSlicHdr);
2322         if (ACPI_FAILURE (Status))
2323         {
2324             return;
2325         }

2327         switch (SubTable->Type)
2328         {
2329         case ACPI_SLIC_TYPE_PUBLIC_KEY:

2331             InfoTable = AcpiDmTableInfoSlic0;
2332             break;

2334         case ACPI_SLIC_TYPE_WINDOWS_MARKER:

2336             InfoTable = AcpiDmTableInfoSlic1;
2337             break;

2339         default:

2341             AcpiOsPrintf ("\n**** Unknown SLIC sub-table type 0x%X\n", SubTable-

2343             /* Attempt to continue */

2345             if (!SubTable->Length)
2346             {
2347                 AcpiOsPrintf ("Invalid zero length subtable\n");
2348                 return;
2349             }
2350             goto NextSubTable;
2351         }

2353         AcpiOsPrintf ("\n");
2354         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2355                     SubTable->Length, InfoTable);
2356         if (ACPI_FAILURE (Status))
2357         {
2358             return;
2359         }

2361 NextSubTable:
2362         /* Point to next sub-table */

2364         Offset += SubTable->Length;
2365         SubTable = ACPI_ADD_PTR (ACPI_SLIC_HEADER, SubTable, SubTable->Length);
2366     }
2367 }
```

```
2370 /*******************************************************************************
2371  *
2372  * FUNCTION:    AcpiDmDumpSlit
2373  *
2374  * PARAMETERS:  Table               - An SLIT
2375  *
2376  * RETURN:      None
2377  *
2378  * DESCRIPTION: Format the contents of a SLIT
2379  *
2380  ******************************************************************************/

2382 void
2383 AcpiDmDumpSlit (
2384     ACPI_TABLE_HEADER       *Table)
2385 {
2386     ACPI_STATUS             Status;
2387     UINT32                  Offset;
2388     UINT8                   *Row;
2389     UINT32                  Localities;
2390     UINT32                  i;
2391     UINT32                  j;


2394     /* Main table */

2396     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoSlit);
2397     if (ACPI_FAILURE (Status))
2398     {
2399         return;
2400     }

2402     /* Display the Locality NxN Matrix */

2404     Localities = (UINT32) ACPI_CAST_PTR (ACPI_TABLE_SLIT, Table)->LocalityCount;
2405     Offset = ACPI_OFFSET (ACPI_TABLE_SLIT, Entry[0]);
2406     Row = (UINT8 *) ACPI_CAST_PTR (ACPI_TABLE_SLIT, Table)->Entry;

2408     for (i = 0; i < Localities; i++)
2409     {
2410         /* Display one row of the matrix */

2412         AcpiDmLineHeader2 (Offset, Localities, "Locality", i);
2413         for  (j = 0; j < Localities; j++)
2414         {
2415             /* Check for beyond EOT */

2417             if (Offset >= Table->Length)
2418             {
2419                 AcpiOsPrintf ("\n**** Not enough room in table for all localitie
2420                 return;
2421             }

2423             AcpiOsPrintf ("%2.2X", Row[j]);
2424             Offset++;

2426             /* Display up to 16 bytes per output row */

2428             if ((j+1) < Localities)
2429             {
2430                 AcpiOsPrintf (" ");

2432                 if (j && (((j+1) % 16) == 0))
2433                 {
2434                     AcpiOsPrintf ("\\\n"); /* With line continuation char */
2435                     AcpiDmLineHeader (Offset, 0, NULL);
```



```
2436                 }
2437             }
2438         }

2440         /* Point to next row */

2442         AcpiOsPrintf ("\n");
2443         Row += Localities;
2444     }
2445 }


2448 /*******************************************************************************
2449  *
2450  * FUNCTION:    AcpiDmDumpSrat
2451  *
2452  * PARAMETERS:  Table               - A SRAT table
2453  *
2454  * RETURN:      None
2455  *
2456  * DESCRIPTION: Format the contents of a SRAT
2457  *
2458  ******************************************************************************/

2460 void
2461 AcpiDmDumpSrat (
2462     ACPI_TABLE_HEADER       *Table)
2463 {
2464     ACPI_STATUS             Status;
2465     UINT32                  Offset = sizeof (ACPI_TABLE_SRAT);
2466     ACPI_SUBTABLE_HEADER    *SubTable;
2467     ACPI_DMTABLE_INFO       *InfoTable;


2470     /* Main table */

2472     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoSrat);
2473     if (ACPI_FAILURE (Status))
2474     {
2475         return;
2476     }

2478     /* Sub-tables */

2480     SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, Table, Offset);
2481     while (Offset < Table->Length)
2482     {
2483         /* Common sub-table header */

2485         AcpiOsPrintf ("\n");
2486         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2487                     SubTable->Length, AcpiDmTableInfoSratHdr);
2488         if (ACPI_FAILURE (Status))
2489         {
2490             return;
2491         }

2493         switch (SubTable->Type)
2494         {
2495         case ACPI_SRAT_TYPE_CPU_AFFINITY:

2497             InfoTable = AcpiDmTableInfoSrat0;
2498             break;

2500         case ACPI_SRAT_TYPE_MEMORY_AFFINITY:
```

```
2502             InfoTable = AcpiDmTableInfoSrat1;
2503             break;

2505         case ACPI_SRAT_TYPE_X2APIC_CPU_AFFINITY:

2507             InfoTable = AcpiDmTableInfoSrat2;
2508             break;

2510         default:
2511             AcpiOsPrintf ("\n**** Unknown SRAT sub-table type 0x%X\n", SubTable-

2513             /* Attempt to continue */

2515             if (!SubTable->Length)
2516             {
2517                 AcpiOsPrintf ("Invalid zero length subtable\n");
2518                 return;
2519             }
2520             goto NextSubTable;
2521         }

2523         AcpiOsPrintf ("\n");
2524         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2525                     SubTable->Length, InfoTable);
2526         if (ACPI_FAILURE (Status))
2527         {
2528             return;
2529         }

2531 NextSubTable:
2532         /* Point to next sub-table */

2534         Offset += SubTable->Length;
2535         SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, SubTable, SubTable->Lengt
2536     }
2537 }


2540 /******************************************************************************
2541  *
2542  * FUNCTION:    AcpiDmDumpVrtc
2543  *
2544  * PARAMETERS:  Table               - A VRTC table
2545  *
2546  * RETURN:      None
2547  *
2548  * DESCRIPTION: Format the contents of a VRTC
2549  *
2550  *****************************************************************************/

2552 void
2553 AcpiDmDumpVrtc (
2554     ACPI_TABLE_HEADER       *Table)
2555 {
2556     ACPI_STATUS             Status;
2557     UINT32                  Offset = sizeof (ACPI_TABLE_VRTC);
2558     ACPI_VRTC_ENTRY         *SubTable;


2561     /* Main table */

2563     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoVrtc);
2564     if (ACPI_FAILURE (Status))
2565     {
2566         return;
2567     }
```



```
2569     /* Sub-tables */

2571     SubTable = ACPI_ADD_PTR (ACPI_VRTC_ENTRY, Table, Offset);
2572     while (Offset < Table->Length)
2573     {
2574         /* Common sub-table header */

2576         AcpiOsPrintf ("\n");
2577         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2578                     sizeof (ACPI_VRTC_ENTRY), AcpiDmTableInfoVrtc0);
2579         if (ACPI_FAILURE (Status))
2580         {
2581             return;
2582         }

2584         /* Point to next sub-table */

2586         Offset += sizeof (ACPI_VRTC_ENTRY);
2587         SubTable = ACPI_ADD_PTR (ACPI_VRTC_ENTRY, SubTable, sizeof (ACPI_VRTC_EN
2588     }
2589 }


2592 /******************************************************************************
2593  *
2594  * FUNCTION:    AcpiDmDumpWdat
2595  *
2596  * PARAMETERS:  Table               - A WDAT table
2597  *
2598  * RETURN:      None
2599  *
2600  * DESCRIPTION: Format the contents of a WDAT
2601  *
2602  *****************************************************************************/

2604 void
2605 AcpiDmDumpWdat (
2606     ACPI_TABLE_HEADER       *Table)
2607 {
2608     ACPI_STATUS             Status;
2609     UINT32                  Offset = sizeof (ACPI_TABLE_WDAT);
2610     ACPI_WDAT_ENTRY         *SubTable;


2613     /* Main table */

2615     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoWdat);
2616     if (ACPI_FAILURE (Status))
2617     {
2618         return;
2619     }

2621     /* Sub-tables */

2623     SubTable = ACPI_ADD_PTR (ACPI_WDAT_ENTRY, Table, Offset);
2624     while (Offset < Table->Length)
2625     {
2626         /* Common sub-table header */

2628         AcpiOsPrintf ("\n");
2629         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
2630                     sizeof (ACPI_WDAT_ENTRY), AcpiDmTableInfoWdat0);
2631         if (ACPI_FAILURE (Status))
2632         {
2633             return;
```

```
2634         }
2636         /* Point to next sub-table */
2638         Offset += sizeof (ACPI_WDAT_ENTRY);
2639         SubTable = ACPI_ADD_PTR (ACPI_WDAT_ENTRY, SubTable, sizeof (ACPI_WDAT_EN
2640     }
2641 }
```

```
**********************************************************
  109757 Thu Dec 26 13:48:25 2013
new/usr/src/common/acpica/common/dmtbinfo.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmtbinfo - Table info for non-AML tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"

/* This module used for application-level code only */

#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmtbinfo")

/*
 * How to add a new table:
 *
 * - Add the C table definition to the actbl1.h or actbl2.h header.
 * - Add ACPI_xxxx_OFFSET macro(s) for the table (and subtables) to list below.
 * - Define the table in this file (for the disassembler). If any
 *   new data types are required (ACPI_DMT_*), see below.
```



```
 * - Add an external declaration for the new table definition (AcpiDmTableInfo*)
 *     in acdisam.h
 * - Add new table definition to the dispatch table in dmtable.c (AcpiDmTableDat
 *     If a simple table (with no subtables), no disassembly code is needed.
 *     Otherwise, create the AcpiDmDump* function for to disassemble the table
 *     and add it to the dmtbdump.c file.
 * - Add an external declaration for the new AcpiDmDump* function in acdisasm.h
 * - Add the new AcpiDmDump* function to the dispatch table in dmtable.c
 * - Create a template for the new table
 * - Add data table compiler support
 *
 * How to add a new data type (ACPI_DMT_*):
 *
 * - Add new type at the end of the ACPI_DMT list in acdisasm.h
 * - Add length and implementation cases in dmtable.c  (disassembler)
 * - Add type and length cases in dtutils.c (DT compiler)
 */

/*
 * Macros used to generate offsets to specific table fields
 */
#define ACPI_FACS_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_FACS,f)
#define ACPI_GAS_OFFSET(f)              (UINT16) ACPI_OFFSET (ACPI_GENERIC_ADDRE
#define ACPI_HDR_OFFSET(f)              (UINT16) ACPI_OFFSET (ACPI_TABLE_HEADER,
#define ACPI_RSDP_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_RSDP,f)
#define ACPI_BERT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_BERT,f)
#define ACPI_BGRT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_BGRT,f)
#define ACPI_BOOT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_BOOT,f)
#define ACPI_CPEP_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_CPEP,f)
#define ACPI_DBG2_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_DBG2,f)
#define ACPI_DBGP_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_DBGP,f)
#define ACPI_DMAR_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_DMAR,f)
#define ACPI_DRTM_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_DRTM,f)
#define ACPI_ECDT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_ECDT,f)
#define ACPI_EINJ_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_EINJ,f)
#define ACPI_ERST_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_ERST,f)
#define ACPI_GTDT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_GTDT,f)
#define ACPI_HEST_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_HEST,f)
#define ACPI_HPET_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_HPET,f)
#define ACPI_IVRS_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_IVRS,f)
#define ACPI_MADT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_MADT,f)
#define ACPI_MCFG_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_MCFG,f)
#define ACPI_MCHI_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_MCHI,f)
#define ACPI_MPST_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_MPST,f)
#define ACPI_MSCT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_MSCT,f)
#define ACPI_PCCT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_PCCT,f)
#define ACPI_PMTT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_PMTT,f)
#define ACPI_S3PT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_S3PT,f)
#define ACPI_SBST_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_SBST,f)
#define ACPI_SLIT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_SLIT,f)
#define ACPI_SPCR_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_SPCR,f)
#define ACPI_SPMI_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_SPMI,f)
#define ACPI_SRAT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_SRAT,f)
#define ACPI_TCPA_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_TCPA,f)
#define ACPI_TPM2_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_TPM2,f)
#define ACPI_UEFI_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_UEFI,f)
#define ACPI_WAET_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_WAET,f)
#define ACPI_WDAT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_WDAT,f)
#define ACPI_WDDT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_WDDT,f)
#define ACPI_WDRT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_WDRT,f)

/* Subtables */

#define ACPI_ASF0_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_ASF_INFO,f)
#define ACPI_ASF1_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_ASF_ALERT,f)
#define ACPI_ASF1a_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_ASF_ALERT_DAT
```

```
126 #define ACPI_ASF2_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_ASF_REMOTE,f)
127 #define ACPI_ASF2a_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_ASF_CONTROL_D
128 #define ACPI_ASF3_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_ASF_RMCP,f)
129 #define ACPI_ASF4_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_ASF_ADDRESS,f
130 #define ACPI_CPEP0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_CPEP_POLLING,
131 #define ACPI_CSRT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_CSRT_GROUP,f)
132 #define ACPI_CSRT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_CSRT_SHARED_I
133 #define ACPI_CSRT2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_CSRT_DESCRIPT
134 #define ACPI_DBG20_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DBG2_DEVICE,f
135 #define ACPI_DMARS_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DMAR_DEVICE_S
136 #define ACPI_DMAR0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DMAR_HARDWARE
137 #define ACPI_DMAR1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DMAR_RESERVED
138 #define ACPI_DMAR2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DMAR_ATSR,f)
139 #define ACPI_DMAR3_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_DMAR_RHSA,f)
140 #define ACPI_EINJ0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_WHEA_HEADER,f
141 #define ACPI_ERST0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_WHEA_HEADER,f
142 #define ACPI_FPDTH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_FPDT_HEADER,f
143 #define ACPI_FPDT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_FPDT_BOOT,f)
144 #define ACPI_FPDT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_FPDT_S3PT_PTR
145 #define ACPI_HEST0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_IA_MACHI
146 #define ACPI_HEST1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_IA_CORRE
147 #define ACPI_HEST2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_IA_NMI,f
148 #define ACPI_HEST6_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_AER_ROOT
149 #define ACPI_HEST7_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_AER,f)
150 #define ACPI_HEST8_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_AER_BRID
151 #define ACPI_HEST9_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_GENERIC,
152 #define ACPI_HESTN_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_NOTIFY,f
153 #define ACPI_HESTB_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_HEST_IA_ERROR
154 #define ACPI_IVRSH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_IVRS_HEADER,f
155 #define ACPI_IVRS0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_IVRS_HARDWARE
156 #define ACPI_IVRS1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_IVRS_MEMORY,f
157 #define ACPI_IVRSD_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_IVRS_DE_HEADE
158 #define ACPI_IVRS8A_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_IVRS_DEVICE8A
159 #define ACPI_IVRS8B_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_IVRS_DEVICE8B
160 #define ACPI_IVRS8C_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_IVRS_DEVICE8C
161 #define ACPI_MADT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_AP
162 #define ACPI_MADT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_IO_APIC,
163 #define ACPI_MADT2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_INTERRUP
164 #define ACPI_MADT3_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_NMI_SOUR
165 #define ACPI_MADT4_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_AP
166 #define ACPI_MADT5_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_AP
167 #define ACPI_MADT6_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_IO_SAPIC
168 #define ACPI_MADT7_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_SA
169 #define ACPI_MADT8_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_INTERRUP
170 #define ACPI_MADT9_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_X2
171 #define ACPI_MADT10_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_MADT_LOCAL_X2
172 #define ACPI_MADT11_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_MADT_GENERIC_
173 #define ACPI_MADT12_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_MADT_GENERIC_
174 #define ACPI_MADTH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SUBTABLE_HEAD
175 #define ACPI_MCFG0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MCFG_ALLOCATI
176 #define ACPI_MPST0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MPST_POWER_NO
177 #define ACPI_MPST0A_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_MPST_POWER_ST
178 #define ACPI_MPST0B_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_MPST_COMPONEN
179 #define ACPI_MPST1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MPST_DATA_HDR
180 #define ACPI_MPST2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MPST_POWER_DA
181 #define ACPI_MSCT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MSCT_PROXIMIT
182 #define ACPI_MTMR0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_MTMR_ENTRY,f)
183 #define ACPI_PCCT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_PCCT_SUBSPACE
184 #define ACPI_PMTT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_PMTT_SOCKET,f
185 #define ACPI_PMTT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_PMTT_CONTROLL
186 #define ACPI_PMTT1A_OFFSET(f)           (UINT16) ACPI_OFFSET (ACPI_PMTT_DOMAIN,f
187 #define ACPI_PMTT2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_PMTT_PHYSICAL
188 #define ACPI_PMTTH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_PMTT_HEADER,f
189 #define ACPI_S3PTH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_S3PT_HEADER,f
190 #define ACPI_S3PT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_S3PT_RESUME,f
191 #define ACPI_S3PT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_S3PT_SUSPEND,
```



```
192 #define ACPI_SLICH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SLIC_HEADER,f
193 #define ACPI_SLIC0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SLIC_KEY,f)
194 #define ACPI_SLIC1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SLIC_MARKER,f
195 #define ACPI_SRATH_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SUBTABLE_HEAD
196 #define ACPI_SRAT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SRAT_CPU_AFFI
197 #define ACPI_SRAT1_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SRAT_MEM_AFFI
198 #define ACPI_SRAT2_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_SRAT_X2APIC_C
199 #define ACPI_VRTC0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_VRTC_ENTRY,f)
200 #define ACPI_WDAT0_OFFSET(f)            (UINT16) ACPI_OFFSET (ACPI_WDAT_ENTRY,f)

202 /*
203  * Simplify access to flag fields by breaking them up into bytes
204  */
205 #define ACPI_FLAG_OFFSET(d,f,o)         (UINT16) (ACPI_OFFSET (d,f) + o)

207 /* Flags */

209 #define ACPI_FADT_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_FADT,f,o)
210 #define ACPI_FACS_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_FACS,f,o)
211 #define ACPI_HPET_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_HPET,f,o)
212 #define ACPI_SRAT0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_SRAT_CPU_AFFINITY
213 #define ACPI_SRAT1_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_SRAT_MEM_AFFINITY
214 #define ACPI_SRAT2_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_SRAT_X2APIC_CPU_A
215 #define ACPI_GTDT_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_GTDT,f,o)
216 #define ACPI_MADT_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_MADT,f,o)
217 #define ACPI_MADT0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_LOCAL_APIC,f
218 #define ACPI_MADT2_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_INTERRUPT_OV
219 #define ACPI_MADT3_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_NMI_SOURCE,f
220 #define ACPI_MADT4_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_LOCAL_APIC_N
221 #define ACPI_MADT7_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_LOCAL_SAPIC,
222 #define ACPI_MADT8_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_INTERRUPT_SO
223 #define ACPI_MADT9_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MADT_LOCAL_X2APIC
224 #define ACPI_MADT10_FLAG_OFFSET(f,o)    ACPI_FLAG_OFFSET (ACPI_MADT_LOCAL_X2APIC
225 #define ACPI_MADT11_FLAG_OFFSET(f,o)    ACPI_FLAG_OFFSET (ACPI_MADT_GENERIC_INTE
226 #define ACPI_MPST0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MPST_POWER_NODE,f
227 #define ACPI_MPST2_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_MPST_POWER_DATA,f
228 #define ACPI_PCCT_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_PCCT,f,o)
229 #define ACPI_PMTTH_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_PMTT_HEADER,f,o)
230 #define ACPI_WDDT_FLAG_OFFSET(f,o)      ACPI_FLAG_OFFSET (ACPI_TABLE_WDDT,f,o)
231 #define ACPI_EINJ0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_WHEA_HEADER,f,o)
232 #define ACPI_ERST0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_WHEA_HEADER,f,o)
233 #define ACPI_HEST0_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_HEST_IA_MACHINE_C
234 #define ACPI_HEST1_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_HEST_IA_CORRECTED
235 #define ACPI_HEST6_FLAG_OFFSET(f,o)     ACPI_FLAG_OFFSET (ACPI_HEST_AER_ROOT,f,o

237 /*
238  * Required terminator for all tables below
239  */
240 #define ACPI_DMT_TERMINATOR             {ACPI_DMT_EXIT, 0, NULL, 0}
241 #define ACPI_DMT_NEW_LINE               {ACPI_DMT_EXTRA_TEXT, 0, "\n", 0}


244 /*
245  * ACPI Table Information, used to dump formatted ACPI tables
246  *
247  * Each entry is of the form:  <Field Type, Field Offset, Field Name>
248  */

250 /******************************************************************************
251  *
252  * Common ACPI table header
253  *
254  *****************************************************************************/

256 ACPI_DMTABLE_INFO           AcpiDmTableInfoHeader[] =
257 {
```

```
258     {ACPI_DMT_SIG,      ACPI_HDR_OFFSET (Signature[0]),            "Signature",
259     {ACPI_DMT_UINT32,   ACPI_HDR_OFFSET (Length),                 "Table Lengt
260     {ACPI_DMT_UINT8,    ACPI_HDR_OFFSET (Revision),               "Revision",
261     {ACPI_DMT_CHKSUM,   ACPI_HDR_OFFSET (Checksum),               "Checksum",
262     {ACPI_DMT_NAME6,    ACPI_HDR_OFFSET (OemId[0]),               "Oem ID", 0}
263     {ACPI_DMT_NAME8,    ACPI_HDR_OFFSET (OemTableId[0]),          "Oem Table I
264     {ACPI_DMT_UINT32,   ACPI_HDR_OFFSET (OemRevision),            "Oem Revisio
265     {ACPI_DMT_NAME4,    ACPI_HDR_OFFSET (AslCompilerId[0]),       "Asl Compile
266     {ACPI_DMT_UINT32,   ACPI_HDR_OFFSET (AslCompilerRevision),    "Asl Compile
267     ACPI_DMT_TERMINATOR
268 };


271 /*******************************************************************************
272  *
273  * GAS - Generic Address Structure
274  *
275  ******************************************************************************/

277 ACPI_DMTABLE_INFO           AcpiDmTableInfoGas[] =
278 {
279     {ACPI_DMT_SPACEID,  ACPI_GAS_OFFSET (SpaceId),                "Space ID",
280     {ACPI_DMT_UINT8,    ACPI_GAS_OFFSET (BitWidth),               "Bit Width",
281     {ACPI_DMT_UINT8,    ACPI_GAS_OFFSET (BitOffset),              "Bit Offset"
282     {ACPI_DMT_ACCWIDTH, ACPI_GAS_OFFSET (AccessWidth),            "Encoded Acc
283     {ACPI_DMT_UINT64,   ACPI_GAS_OFFSET (Address),                "Address", 0
284     ACPI_DMT_TERMINATOR
285 };


288 /*******************************************************************************
289  *
290  * RSDP - Root System Description Pointer (Signature is "RSD PTR ")
291  *
292  ******************************************************************************/

294 ACPI_DMTABLE_INFO           AcpiDmTableInfoRsdp1[] =
295 {
296     {ACPI_DMT_NAME8,    ACPI_RSDP_OFFSET (Signature[0]),           "Signature",
297     {ACPI_DMT_UINT8,    ACPI_RSDP_OFFSET (Checksum),              "Checksum",
298     {ACPI_DMT_NAME6,    ACPI_RSDP_OFFSET (OemId[0]),              "Oem ID", 0}
299     {ACPI_DMT_UINT8,    ACPI_RSDP_OFFSET (Revision),              "Revision",
300     {ACPI_DMT_UINT32,   ACPI_RSDP_OFFSET (RsdtPhysicalAddress),   "RSDT Addres
301     ACPI_DMT_TERMINATOR
302 };

304 /* ACPI 2.0+ Extensions */

306 ACPI_DMTABLE_INFO           AcpiDmTableInfoRsdp2[] =
307 {
308     {ACPI_DMT_UINT32,   ACPI_RSDP_OFFSET (Length),                "Length", DT
309     {ACPI_DMT_UINT64,   ACPI_RSDP_OFFSET (XsdtPhysicalAddress),   "XSDT Addres
310     {ACPI_DMT_UINT8,    ACPI_RSDP_OFFSET (ExtendedChecksum),      "Extended Ch
311     {ACPI_DMT_UINT24,   ACPI_RSDP_OFFSET (Reserved[0]),           "Reserved",
312     ACPI_DMT_TERMINATOR
313 };


316 /*******************************************************************************
317  *
318  * FACS - Firmware ACPI Control Structure
319  *
320  ******************************************************************************/

322 ACPI_DMTABLE_INFO           AcpiDmTableInfoFacs[] =
323 {
```



```
324     {ACPI_DMT_NAME4,    ACPI_FACS_OFFSET (Signature[0]),           "Signature",
325     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (Length),                "Length", DT
326     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (HardwareSignature),     "Hardware Si
327     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (FirmwareWakingVector),  "32 Firmware
328     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (GlobalLock),            "Global Lock
329     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (Flags),                 "Flags (deco
330     {ACPI_DMT_FLAG0,    ACPI_FACS_FLAG_OFFSET (Flags,0),          "S4BIOS Supp
331     {ACPI_DMT_FLAG1,    ACPI_FACS_FLAG_OFFSET (Flags,0),          "64-bit Wake
332     {ACPI_DMT_UINT64,   ACPI_FACS_OFFSET (XFirmwareWakingVector), "64 Firmware
333     {ACPI_DMT_UINT8,    ACPI_FACS_OFFSET (Version),               "Version", 0
334     {ACPI_DMT_UINT24,   ACPI_FACS_OFFSET (Reserved[0]),           "Reserved",
335     {ACPI_DMT_UINT32,   ACPI_FACS_OFFSET (OspmFlags),             "OspmFlags (
336     {ACPI_DMT_FLAG0,    ACPI_FACS_FLAG_OFFSET (OspmFlags,0),      "64-bit Wake
337     ACPI_DMT_TERMINATOR
338 };


341 /*******************************************************************************
342  *
343  * FADT - Fixed ACPI Description Table (Signature is FACP)
344  *
345  ******************************************************************************/

347 /* ACPI 1.0 FADT (Version 1) */

349 ACPI_DMTABLE_INFO           AcpiDmTableInfoFadt1[] =
350 {
351     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Facs),                  "FACS Addres
352     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Dsdt),                  "DSDT Addres
353     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Model),                 "Model", 0},
354     {ACPI_DMT_FADTPM,   ACPI_FADT_OFFSET (PreferredProfile),      "PM Profile"
355     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (SciInterrupt),          "SCI Interru
356     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (SmiCommand),            "SMI Command
357     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (AcpiEnable),            "ACPI Enable
358     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (AcpiDisable),           "ACPI Disabl
359     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (S4BiosRequest),         "S4BIOS Comm
360     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (PstateControl),         "P-State Con
361     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Pm1aEventBlock),        "PM1A Event
362     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Pm1bEventBlock),        "PM1B Event
363     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Pm1aControlBlock),      "PM1A Contro
364     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Pm1bControlBlock),      "PM1B Contro
365     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Pm2ControlBlock),       "PM2 Control
366     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (PmTimerBlock),          "PM Timer Bl
367     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Gpe0Block),             "GPE0 Block
368     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Gpe1Block),             "GPE1 Block
369     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Pm1EventLength),        "PM1 Event B
370     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Pm1ControlLength),      "PM1 Control
371     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Pm2ControlLength),      "PM2 Control
372     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (PmTimerLength),         "PM Timer Bl
373     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Gpe0BlockLength),       "GPE0 Block
374     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Gpe1BlockLength),       "GPE1 Block
375     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Gpe1Base),              "GPE1 Base O
376     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (CstControl),            "_CST Suppor
377     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (C2Latency),             "C2 Latency"
378     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (C3Latency),             "C3 Latency"
379     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (FlushSize),             "CPU Cache S
380     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (FlushStride),           "Cache Flush
381     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (DutyOffset),            "Duty Cycle
382     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (DutyWidth),             "Duty Cycle
383     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (DayAlarm),              "RTC Day Ala
384     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (MonthAlarm),            "RTC Month A
385     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Century),               "RTC Century
386     {ACPI_DMT_UINT16,   ACPI_FADT_OFFSET (BootFlags),             "Boot Flags

388     /* Boot Architecture Flags byte 0 */
```

```
390     {ACPI_DMT_FLAG0,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "Legacy Devi
391     {ACPI_DMT_FLAG1,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "8042 Presen
392     {ACPI_DMT_FLAG2,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "VGA Not Pre
393     {ACPI_DMT_FLAG3,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "MSI Not Sup
394     {ACPI_DMT_FLAG4,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "PCIe ASPM N
395     {ACPI_DMT_FLAG5,    ACPI_FADT_FLAG_OFFSET (BootFlags,0),        "CMOS RTC No

397     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (Reserved),                "Reserved",
398     {ACPI_DMT_UINT32,   ACPI_FADT_OFFSET (Flags),                   "Flags (deco

400     /* Flags byte 0 */

402     {ACPI_DMT_FLAG0,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "WBINVD inst
403     {ACPI_DMT_FLAG1,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "WBINVD flus
404     {ACPI_DMT_FLAG2,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "All CPUs su
405     {ACPI_DMT_FLAG3,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "C2 works on
406     {ACPI_DMT_FLAG4,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "Control Met
407     {ACPI_DMT_FLAG5,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "Control Met
408     {ACPI_DMT_FLAG6,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "RTC wake no
409     {ACPI_DMT_FLAG7,    ACPI_FADT_FLAG_OFFSET (Flags,0),            "RTC can wak

411     /* Flags byte 1 */

413     {ACPI_DMT_FLAG0,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "32-bit PM T
414     {ACPI_DMT_FLAG1,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Docking Sup
415     {ACPI_DMT_FLAG2,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Reset Regis
416     {ACPI_DMT_FLAG3,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Sealed Case
417     {ACPI_DMT_FLAG4,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Headless -
418     {ACPI_DMT_FLAG5,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Use native
419     {ACPI_DMT_FLAG6,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "PCIEXP_WAK
420     {ACPI_DMT_FLAG7,    ACPI_FADT_FLAG_OFFSET (Flags,1),            "Use Platfor

422     /* Flags byte 2 */

424     {ACPI_DMT_FLAG0,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "RTC_STS val
425     {ACPI_DMT_FLAG1,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "Remote Powe
426     {ACPI_DMT_FLAG2,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "Use APIC Cl
427     {ACPI_DMT_FLAG3,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "Use APIC Ph
428     {ACPI_DMT_FLAG4,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "Hardware Re
429     {ACPI_DMT_FLAG5,    ACPI_FADT_FLAG_OFFSET (Flags,2),            "Low Power S
430     ACPI_DMT_TERMINATOR
431 };

433 /* ACPI 1.0 MS Extensions (FADT version 2) */

435 ACPI_DMTABLE_INFO           AcpiDmTableInfoFadt2[] =
436 {
437     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (ResetRegister),           "Reset Regis
438     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (ResetValue),              "Value to ca
439     {ACPI_DMT_UINT24,   ACPI_FADT_OFFSET (Reserved4[0]),            "Reserved",
440     ACPI_DMT_TERMINATOR
441 };

443 /* ACPI 2.0+ Extensions (FADT version 3 and 4) */

445 ACPI_DMTABLE_INFO           AcpiDmTableInfoFadt3[] =
446 {
447     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (ResetRegister),           "Reset Regis
448     {ACPI_DMT_UINT8,    ACPI_FADT_OFFSET (ResetValue),              "Value to ca
449     {ACPI_DMT_UINT24,   ACPI_FADT_OFFSET (Reserved4[0]),            "Reserved",
450     {ACPI_DMT_UINT64,   ACPI_FADT_OFFSET (XFacs),                   "FACS Addres
451     {ACPI_DMT_UINT64,   ACPI_FADT_OFFSET (XDsdt),                   "DSDT Addres
452     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPm1aEventBlock),         "PM1A Event
453     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPm1bEventBlock),         "PM1B Event
454     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPm1aControlBlock),       "PM1A Contro
455     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPm1bControlBlock),       "PM1B Contro
```



```
456     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPm2ControlBlock),        "PM2 Control
457     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XPmTimerBlock),           "PM Timer Bl
458     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XGpe0Block),              "GPE0 Block"
459     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (XGpe1Block),              "GPE1 Block"
460     ACPI_DMT_TERMINATOR
461 };

463 /* ACPI 5.0 Extensions (FADT version 5) */

465 ACPI_DMTABLE_INFO           AcpiDmTableInfoFadt5[] =
466 {
467     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (SleepControl),            "Sleep Contr
468     {ACPI_DMT_GAS,      ACPI_FADT_OFFSET (SleepStatus),             "Sleep Statu
469     ACPI_DMT_TERMINATOR
470 };


473 /*
474  * Remaining tables are not consumed directly by the ACPICA subsystem
475  */

477 /*******************************************************************************
478  *
479  * ASF - Alert Standard Format table (Signature "ASF!")
480  *
481  ******************************************************************************/

483 /* Common Subtable header (one per Subtable) */

485 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsfHdr[] =
486 {
487     {ACPI_DMT_ASF,      ACPI_ASF0_OFFSET (Header.Type),             "Subtable Ty
488     {ACPI_DMT_UINT8,    ACPI_ASF0_OFFSET (Header.Reserved),         "Reserved",
489     {ACPI_DMT_UINT16,   ACPI_ASF0_OFFSET (Header.Length),           "Length", DT
490     ACPI_DMT_TERMINATOR
491 };

493 /* 0: ASF Information */

495 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf0[] =
496 {
497     {ACPI_DMT_UINT8,    ACPI_ASF0_OFFSET (MinResetValue),           "Minimum Res
498     {ACPI_DMT_UINT8,    ACPI_ASF0_OFFSET (MinPollInterval),         "Minimum Pol
499     {ACPI_DMT_UINT16,   ACPI_ASF0_OFFSET (SystemId),                "System ID",
500     {ACPI_DMT_UINT32,   ACPI_ASF0_OFFSET (MfgId),                   "Manufacture
501     {ACPI_DMT_UINT8,    ACPI_ASF0_OFFSET (Flags),                   "Flags", 0},
502     {ACPI_DMT_UINT24,   ACPI_ASF0_OFFSET (Reserved2[0]),            "Reserved",
503     ACPI_DMT_TERMINATOR
504 };

506 /* 1: ASF Alerts */

508 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf1[] =
509 {
510     {ACPI_DMT_UINT8,    ACPI_ASF1_OFFSET (AssertMask),              "AssertMask"
511     {ACPI_DMT_UINT8,    ACPI_ASF1_OFFSET (DeassertMask),            "DeassertMas
512     {ACPI_DMT_UINT8,    ACPI_ASF1_OFFSET (Alerts),                  "Alert Count
513     {ACPI_DMT_UINT8,    ACPI_ASF1_OFFSET (DataLength),              "Alert Data
514     ACPI_DMT_TERMINATOR
515 };

517 /* 1a: ASF Alert data */

519 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf1a[] =
520 {
521     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Address),                "Address", 0
```

```
522     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Command),                "Command", 0
523     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Mask),                   "Mask", 0},
524     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Value),                  "Value", 0},
525     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (SensorType),             "SensorType"
526     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Type),                   "Type", 0},
527     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Offset),                 "Offset", 0}
528     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (SourceType),             "SourceType"
529     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Severity),               "Severity",
530     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (SensorNumber),           "SensorNumbe
531     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Entity),                 "Entity", 0}
532     {ACPI_DMT_UINT8,    ACPI_ASF1a_OFFSET (Instance),               "Instance",
533     ACPI_DMT_TERMINATOR
534 };

536 /* 2: ASF Remote Control */

538 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf2[] =
539 {
540     {ACPI_DMT_UINT8,    ACPI_ASF2_OFFSET (Controls),                "Control Cou
541     {ACPI_DMT_UINT8,    ACPI_ASF2_OFFSET (DataLength),              "Control Dat
542     {ACPI_DMT_UINT16,   ACPI_ASF2_OFFSET (Reserved2),               "Reserved",
543     ACPI_DMT_TERMINATOR
544 };

546 /* 2a: ASF Control data */

548 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf2a[] =
549 {
550     {ACPI_DMT_UINT8,    ACPI_ASF2a_OFFSET (Function),               "Function",
551     {ACPI_DMT_UINT8,    ACPI_ASF2a_OFFSET (Address),                "Address", 0
552     {ACPI_DMT_UINT8,    ACPI_ASF2a_OFFSET (Command),                "Command", 0
553     {ACPI_DMT_UINT8,    ACPI_ASF2a_OFFSET (Value),                  "Value", 0},
554     ACPI_DMT_TERMINATOR
555 };

557 /* 3: ASF RMCP Boot Options */

559 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf3[] =
560 {
561     {ACPI_DMT_BUF7,     ACPI_ASF3_OFFSET (Capabilities[0]),         "Capabilitie
562     {ACPI_DMT_UINT8,    ACPI_ASF3_OFFSET (CompletionCode),          "Completion
563     {ACPI_DMT_UINT32,   ACPI_ASF3_OFFSET (EnterpriseId),            "Enterprise
564     {ACPI_DMT_UINT8,    ACPI_ASF3_OFFSET (Command),                 "Command", 0
565     {ACPI_DMT_UINT16,   ACPI_ASF3_OFFSET (Parameter),               "Parameter",
566     {ACPI_DMT_UINT16,   ACPI_ASF3_OFFSET (BootOptions),             "Boot Option
567     {ACPI_DMT_UINT16,   ACPI_ASF3_OFFSET (OemParameters),           "Oem Paramet
568     ACPI_DMT_TERMINATOR
569 };

571 /* 4: ASF Address */

573 ACPI_DMTABLE_INFO           AcpiDmTableInfoAsf4[] =
574 {
575     {ACPI_DMT_UINT8,    ACPI_ASF4_OFFSET (EpromAddress),            "Eprom Addre
576     {ACPI_DMT_UINT8,    ACPI_ASF4_OFFSET (Devices),                 "Device Coun
577     ACPI_DMT_TERMINATOR
578 };


581 /*******************************************************************************
582  *
583  * BERT -  Boot Error Record table
584  *
585  ******************************************************************************/

587 ACPI_DMTABLE_INFO           AcpiDmTableInfoBert[] =
```



```
588 {
589     {ACPI_DMT_UINT32,   ACPI_BERT_OFFSET (RegionLength),            "Boot Error
590     {ACPI_DMT_UINT64,   ACPI_BERT_OFFSET (Address),                 "Boot Error
591     ACPI_DMT_TERMINATOR
592 };


595 /*******************************************************************************
596  *
597  * BGRT -  Boot Graphics Resource Table (ACPI 5.0)
598  *
599  ******************************************************************************/

601 ACPI_DMTABLE_INFO           AcpiDmTableInfoBgrt[] =
602 {
603     {ACPI_DMT_UINT16,   ACPI_BGRT_OFFSET (Version),                 "Version", 0
604     {ACPI_DMT_UINT8,    ACPI_BGRT_OFFSET (Status),                  "Status", 0}
605     {ACPI_DMT_UINT8,    ACPI_BGRT_OFFSET (ImageType),               "Image Type"
606     {ACPI_DMT_UINT64,   ACPI_BGRT_OFFSET (ImageAddress),            "Image Addre
607     {ACPI_DMT_UINT32,   ACPI_BGRT_OFFSET (ImageOffsetX),            "Image Offse
608     {ACPI_DMT_UINT32,   ACPI_BGRT_OFFSET (ImageOffsetY),            "Image Offse
609     ACPI_DMT_TERMINATOR
610 };


613 /*******************************************************************************
614  *
615  * BOOT - Simple Boot Flag Table
616  *
617  ******************************************************************************/

619 ACPI_DMTABLE_INFO           AcpiDmTableInfoBoot[] =
620 {
621     {ACPI_DMT_UINT8,    ACPI_BOOT_OFFSET (CmosIndex),               "Boot Regist
622     {ACPI_DMT_UINT24,   ACPI_BOOT_OFFSET (Reserved[0]),             "Reserved",
623     ACPI_DMT_TERMINATOR
624 };


627 /*******************************************************************************
628  *
629  * CPEP - Corrected Platform Error Polling table
630  *
631  ******************************************************************************/

633 ACPI_DMTABLE_INFO           AcpiDmTableInfoCpep[] =
634 {
635     {ACPI_DMT_UINT64,   ACPI_CPEP_OFFSET (Reserved),                "Reserved",
636     ACPI_DMT_TERMINATOR
637 };

639 ACPI_DMTABLE_INFO           AcpiDmTableInfoCpep0[] =
640 {
641     {ACPI_DMT_UINT8,    ACPI_CPEP0_OFFSET (Header.Type),            "Subtable Ty
642     {ACPI_DMT_UINT8,    ACPI_CPEP0_OFFSET (Header.Length),          "Length", DT
643     {ACPI_DMT_UINT8,    ACPI_CPEP0_OFFSET (Id),                     "Processor I
644     {ACPI_DMT_UINT8,    ACPI_CPEP0_OFFSET (Eid),                    "Processor E
645     {ACPI_DMT_UINT32,   ACPI_CPEP0_OFFSET (Interval),               "Polling Int
646     ACPI_DMT_TERMINATOR
647 };


650 /*******************************************************************************
651  *
652  * CSRT - Core System Resource Table
653  *
```

```
 *******************************************************************************/

/* Main table consists only of the standard ACPI table header */

/* Resource Group subtable */

ACPI_DMTABLE_INFO           AcpiDmTableInfoCsrt0[] =
{
    {ACPI_DMT_UINT32,   ACPI_CSRT0_OFFSET (Length),                 "Length", 0}
    {ACPI_DMT_UINT32,   ACPI_CSRT0_OFFSET (VendorId),               "Vendor ID",
    {ACPI_DMT_UINT32,   ACPI_CSRT0_OFFSET (SubvendorId),            "Subvendor I
    {ACPI_DMT_UINT16,   ACPI_CSRT0_OFFSET (DeviceId),               "Device ID",
    {ACPI_DMT_UINT16,   ACPI_CSRT0_OFFSET (SubdeviceId),            "Subdevice I
    {ACPI_DMT_UINT16,   ACPI_CSRT0_OFFSET (Revision),               "Revision",
    {ACPI_DMT_UINT16,   ACPI_CSRT0_OFFSET (Reserved),               "Reserved",
    {ACPI_DMT_UINT32,   ACPI_CSRT0_OFFSET (SharedInfoLength),       "Shared Info
    ACPI_DMT_TERMINATOR
};

/* Shared Info subtable */

ACPI_DMTABLE_INFO           AcpiDmTableInfoCsrt1[] =
{
    {ACPI_DMT_UINT16,   ACPI_CSRT1_OFFSET (MajorVersion),           "Major Versi
    {ACPI_DMT_UINT16,   ACPI_CSRT1_OFFSET (MinorVersion),           "Minor Versi
    {ACPI_DMT_UINT32,   ACPI_CSRT1_OFFSET (MmioBaseLow),            "MMIO Base A
    {ACPI_DMT_UINT32,   ACPI_CSRT1_OFFSET (MmioBaseHigh),           "MMIO Base A
    {ACPI_DMT_UINT32,   ACPI_CSRT1_OFFSET (GsiInterrupt),           "GSI Interru
    {ACPI_DMT_UINT8,    ACPI_CSRT1_OFFSET (InterruptPolarity),      "Interrupt P
    {ACPI_DMT_UINT8,    ACPI_CSRT1_OFFSET (InterruptMode),          "Interrupt M
    {ACPI_DMT_UINT8,    ACPI_CSRT1_OFFSET (NumChannels),            "Num Channel
    {ACPI_DMT_UINT8,    ACPI_CSRT1_OFFSET (DmaAddressWidth),        "DMA Address
    {ACPI_DMT_UINT16,   ACPI_CSRT1_OFFSET (BaseRequestLine),        "Base Reques
    {ACPI_DMT_UINT16,   ACPI_CSRT1_OFFSET (NumHandshakeSignals),    "Num Handsha
    {ACPI_DMT_UINT32,   ACPI_CSRT1_OFFSET (MaxBlockSize),           "Max Block S
    ACPI_DMT_TERMINATOR
};


/* Resource Descriptor subtable */

ACPI_DMTABLE_INFO           AcpiDmTableInfoCsrt2[] =
{
    {ACPI_DMT_UINT32,   ACPI_CSRT2_OFFSET (Length),                 "Length", 0}
    {ACPI_DMT_UINT16,   ACPI_CSRT2_OFFSET (Type),                   "Type", 0},
    {ACPI_DMT_UINT16,   ACPI_CSRT2_OFFSET (Subtype),                "Subtype", 0
    {ACPI_DMT_UINT32,   ACPI_CSRT2_OFFSET (Uid),                    "UID", 0},
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * DBG2 - Debug Port Table 2
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2[] =
{
    {ACPI_DMT_UINT32,   ACPI_DBG2_OFFSET (InfoOffset),              "Info Offset
    {ACPI_DMT_UINT32,   ACPI_DBG2_OFFSET (InfoCount),               "Info Count"
    ACPI_DMT_TERMINATOR
};

/* Debug Device Information Subtable */
```



```
ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2Device[] =
{
    {ACPI_DMT_UINT8,    ACPI_DBG20_OFFSET (Revision),               "Revision",
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (Length),                 "Length", DT
    {ACPI_DMT_UINT8,    ACPI_DBG20_OFFSET (RegisterCount),          "Register Co
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (NamepathLength),         "Namepath Le
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (NamepathOffset),         "Namepath Of
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (OemDataLength),          "OEM Data Le
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (OemDataOffset),          "OEM Data Of
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (PortType),               "Port Type",
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (PortSubtype),            "Port Subtyp
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (Reserved),               "Reserved",
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (BaseAddressOffset),      "Base Addres
    {ACPI_DMT_UINT16,   ACPI_DBG20_OFFSET (AddressSizeOffset),      "Address Siz
    ACPI_DMT_TERMINATOR
};

/* Variable-length data for the subtable */

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2Addr[] =
{
    {ACPI_DMT_GAS,      0,                                          "Base Addres
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2Size[] =
{
    {ACPI_DMT_UINT32,   0,                                          "Address Siz
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2Name[] =
{
    {ACPI_DMT_STRING,   0,                                          "Namepath",
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbg2OemData[] =
{
    {ACPI_DMT_BUFFER,   0,                                          "OEM Data",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * DBGP - Debug Port
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoDbgp[] =
{
    {ACPI_DMT_UINT8,    ACPI_DBGP_OFFSET (Type),                    "Interface T
    {ACPI_DMT_UINT24,   ACPI_DBGP_OFFSET (Reserved[0]),             "Reserved",
    {ACPI_DMT_GAS,      ACPI_DBGP_OFFSET (DebugPort),               "Debug Port
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * DMAR - DMA Remapping table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoDmar[] =
```

```
786 {
787     {ACPI_DMT_UINT8,    ACPI_DMAR_OFFSET (Width),                   "Host Addres
788     {ACPI_DMT_UINT8,    ACPI_DMAR_OFFSET (Flags),                   "Flags", 0},
789     {ACPI_DMT_BUF10,    ACPI_DMAR_OFFSET (Reserved[0]),             "Reserved",
790     ACPI_DMT_TERMINATOR
791 };

793 /* Common Subtable header (one per Subtable) */

795 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmarHdr[] =
796 {
797     {ACPI_DMT_DMAR,     ACPI_DMAR0_OFFSET (Header.Type),            "Subtable Ty
798     {ACPI_DMT_UINT16,   ACPI_DMAR0_OFFSET (Header.Length),          "Length", DT
799     ACPI_DMT_TERMINATOR
800 };

802 /* Common device scope entry */

804 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmarScope[] =
805 {
806     {ACPI_DMT_UINT8,    ACPI_DMARS_OFFSET (EntryType),              "Device Scop
807     {ACPI_DMT_UINT8,    ACPI_DMARS_OFFSET (Length),                 "Entry Lengt
808     {ACPI_DMT_UINT16,   ACPI_DMARS_OFFSET (Reserved),               "Reserved",
809     {ACPI_DMT_UINT8,    ACPI_DMARS_OFFSET (EnumerationId),          "Enumeration
810     {ACPI_DMT_UINT8,    ACPI_DMARS_OFFSET (Bus),                    "PCI Bus Num
811     ACPI_DMT_TERMINATOR
812 };

814 /* DMAR Subtables */

816 /* 0: Hardware Unit Definition */

818 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmar0[] =
819 {
820     {ACPI_DMT_UINT8,    ACPI_DMAR0_OFFSET (Flags),                  "Flags", 0},
821     {ACPI_DMT_UINT8,    ACPI_DMAR0_OFFSET (Reserved),               "Reserved",
822     {ACPI_DMT_UINT16,   ACPI_DMAR0_OFFSET (Segment),                "PCI Segment
823     {ACPI_DMT_UINT64,   ACPI_DMAR0_OFFSET (Address),                "Register Ba
824     ACPI_DMT_TERMINATOR
825 };

827 /* 1: Reserved Memory Definition */

829 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmar1[] =
830 {
831     {ACPI_DMT_UINT16,   ACPI_DMAR1_OFFSET (Reserved),               "Reserved",
832     {ACPI_DMT_UINT16,   ACPI_DMAR1_OFFSET (Segment),                "PCI Segment
833     {ACPI_DMT_UINT64,   ACPI_DMAR1_OFFSET (BaseAddress),            "Base Addres
834     {ACPI_DMT_UINT64,   ACPI_DMAR1_OFFSET (EndAddress),             "End Address
835     ACPI_DMT_TERMINATOR
836 };

838 /* 2: Root Port ATS Capability Definition */

840 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmar2[] =
841 {
842     {ACPI_DMT_UINT8,    ACPI_DMAR2_OFFSET (Flags),                  "Flags", 0},
843     {ACPI_DMT_UINT8,    ACPI_DMAR2_OFFSET (Reserved),               "Reserved",
844     {ACPI_DMT_UINT16,   ACPI_DMAR2_OFFSET (Segment),                "PCI Segment
845     ACPI_DMT_TERMINATOR
846 };

848 /* 3: Remapping Hardware Static Affinity Structure */

850 ACPI_DMTABLE_INFO           AcpiDmTableInfoDmar3[] =
851 {
```



```
852     {ACPI_DMT_UINT32,   ACPI_DMAR3_OFFSET (Reserved),               "Reserved",
853     {ACPI_DMT_UINT64,   ACPI_DMAR3_OFFSET (BaseAddress),            "Base Addres
854     {ACPI_DMT_UINT32,   ACPI_DMAR3_OFFSET (ProximityDomain),        "Proximity D
855     ACPI_DMT_TERMINATOR
856 };


859 /*******************************************************************************
860  *
861  * DRTM - Dynamic Root of Trust for Measurement table
862  *
863  ******************************************************************************/

865 ACPI_DMTABLE_INFO           AcpiDmTableInfoDrtm[] =
866 {

868     ACPI_DMT_TERMINATOR
869 };


872 /*******************************************************************************
873  *
874  * ECDT - Embedded Controller Boot Resources Table
875  *
876  ******************************************************************************/

878 ACPI_DMTABLE_INFO           AcpiDmTableInfoEcdt[] =
879 {
880     {ACPI_DMT_GAS,      ACPI_ECDT_OFFSET (Control),                 "Command/Sta
881     {ACPI_DMT_GAS,      ACPI_ECDT_OFFSET (Data),                    "Data Regist
882     {ACPI_DMT_UINT32,   ACPI_ECDT_OFFSET (Uid),                     "UID", 0},
883     {ACPI_DMT_UINT8,    ACPI_ECDT_OFFSET (Gpe),                     "GPE Number"
884     {ACPI_DMT_STRING,   ACPI_ECDT_OFFSET (Id[0]),                   "Namepath",
885     ACPI_DMT_TERMINATOR
886 };


889 /*******************************************************************************
890  *
891  * EINJ - Error Injection table
892  *
893  ******************************************************************************/

895 ACPI_DMTABLE_INFO           AcpiDmTableInfoEinj[] =
896 {
897     {ACPI_DMT_UINT32,   ACPI_EINJ_OFFSET (HeaderLength),            "Injection H
898     {ACPI_DMT_UINT8,    ACPI_EINJ_OFFSET (Flags),                   "Flags", 0},
899     {ACPI_DMT_UINT24,   ACPI_EINJ_OFFSET (Reserved[0]),             "Reserved",
900     {ACPI_DMT_UINT32,   ACPI_EINJ_OFFSET (Entries),                 "Injection E
901     ACPI_DMT_TERMINATOR
902 };

904 ACPI_DMTABLE_INFO           AcpiDmTableInfoEinj0[] =
905 {
906     {ACPI_DMT_EINJACT,  ACPI_EINJ0_OFFSET (Action),                 "Action", 0}
907     {ACPI_DMT_EINJINST, ACPI_EINJ0_OFFSET (Instruction),            "Instruction
908     {ACPI_DMT_UINT8,    ACPI_EINJ0_OFFSET (Flags),                  "Flags (deco
909     {ACPI_DMT_FLAG0,    ACPI_EINJ0_FLAG_OFFSET (Flags,0),           "Preserve Re

911     {ACPI_DMT_UINT8,    ACPI_EINJ0_OFFSET (Reserved),               "Reserved",
912     {ACPI_DMT_GAS,      ACPI_EINJ0_OFFSET (RegisterRegion),         "Register Re
913     {ACPI_DMT_UINT64,   ACPI_EINJ0_OFFSET (Value),                  "Value", 0},
914     {ACPI_DMT_UINT64,   ACPI_EINJ0_OFFSET (Mask),                   "Mask", 0},
915     ACPI_DMT_TERMINATOR
916 };
```

```
/*******************************************************************************
 *
 * ERST - Error Record Serialization table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoErst[] =
{
    {ACPI_DMT_UINT32,   ACPI_ERST_OFFSET (HeaderLength),            "Serializati
    {ACPI_DMT_UINT32,   ACPI_ERST_OFFSET (Reserved),                "Reserved",
    {ACPI_DMT_UINT32,   ACPI_ERST_OFFSET (Entries),                 "Instruction
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoErst0[] =
{
    {ACPI_DMT_ERSTACT,  ACPI_ERST0_OFFSET (Action),                 "Action", 0}
    {ACPI_DMT_ERSTINST, ACPI_ERST0_OFFSET (Instruction),            "Instruction
    {ACPI_DMT_UINT8,    ACPI_ERST0_OFFSET (Flags),                  "Flags (deco
    {ACPI_DMT_FLAG0,    ACPI_ERST0_FLAG_OFFSET (Flags,0),           "Preserve Re

    {ACPI_DMT_UINT8,    ACPI_ERST0_OFFSET (Reserved),               "Reserved",
    {ACPI_DMT_GAS,      ACPI_ERST0_OFFSET (RegisterRegion),         "Register Re
    {ACPI_DMT_UINT64,   ACPI_ERST0_OFFSET (Value),                  "Value", 0},
    {ACPI_DMT_UINT64,   ACPI_ERST0_OFFSET (Mask),                   "Mask", 0},
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * FPDT - Firmware Performance Data Table (ACPI 5.0)
 *
 ******************************************************************************/

/* Main table consists of only the standard ACPI header - subtables follow */

/* FPDT subtable header */

ACPI_DMTABLE_INFO           AcpiDmTableInfoFpdtHdr[] =
{
    {ACPI_DMT_UINT16,   ACPI_FPDTH_OFFSET (Type),                   "Subtable Ty
    {ACPI_DMT_UINT8,    ACPI_FPDTH_OFFSET (Length),                 "Length", DT
    {ACPI_DMT_UINT8,    ACPI_FPDTH_OFFSET (Revision),               "Revision",
    ACPI_DMT_TERMINATOR
};

/* 0: Firmware Basic Boot Performance Record */

ACPI_DMTABLE_INFO           AcpiDmTableInfoFpdt0[] =
{
    {ACPI_DMT_UINT32,   ACPI_FPDT0_OFFSET (Reserved),               "Reserved",
    {ACPI_DMT_UINT64,   ACPI_FPDT0_OFFSET (ResetEnd),               "Reset End",
    {ACPI_DMT_UINT64,   ACPI_FPDT0_OFFSET (LoadStart),              "Load Image
    {ACPI_DMT_UINT64,   ACPI_FPDT0_OFFSET (StartupStart),           "Start Image
    {ACPI_DMT_UINT64,   ACPI_FPDT0_OFFSET (ExitServicesEntry),      "Exit Servic
    {ACPI_DMT_UINT64,   ACPI_FPDT0_OFFSET (ExitServicesExit),       "Exit Servic
    ACPI_DMT_TERMINATOR
};

/* 1: S3 Performance Table Pointer Record */

ACPI_DMTABLE_INFO           AcpiDmTableInfoFpdt1[] =
{
    {ACPI_DMT_UINT32,   ACPI_FPDT1_OFFSET (Reserved),               "Reserved",
```



```
    {ACPI_DMT_UINT64,   ACPI_FPDT1_OFFSET (Address),                "S3PT Addres
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * GTDT - Generic Timer Description Table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoGtdt[] =
{
    {ACPI_DMT_UINT64,   ACPI_GTDT_OFFSET (Address),                 "Timer Addre
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (Flags),                   "Flags (deco
    {ACPI_DMT_FLAG0,    ACPI_GTDT_FLAG_OFFSET (Flags,0),            "Memory Pres
    ACPI_DMT_NEW_LINE,
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (SecurePl1Interrupt),      "Secure PL1
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (SecurePl1Flags),          "SPL1 Flags
    {ACPI_DMT_FLAG0,    ACPI_GTDT_FLAG_OFFSET (SecurePl1Flags,0),   "Trigger Mod
    {ACPI_DMT_FLAG1,    ACPI_GTDT_FLAG_OFFSET (SecurePl1Flags,0),   "Polarity",
    ACPI_DMT_NEW_LINE,
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (NonSecurePl1Interrupt),   "Non-Secure
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (NonSecurePl1Flags),       "NSPL1 Flags
    {ACPI_DMT_FLAG0,    ACPI_GTDT_FLAG_OFFSET (NonSecurePl1Flags,0),"Trigger Mod
    {ACPI_DMT_FLAG1,    ACPI_GTDT_FLAG_OFFSET (NonSecurePl1Flags,0),"Polarity",
    ACPI_DMT_NEW_LINE,
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (VirtualTimerInterrupt),   "Virtual Tim
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (VirtualTimerFlags),       "VT Flags (d
    {ACPI_DMT_FLAG0,    ACPI_GTDT_FLAG_OFFSET (VirtualTimerFlags,0),"Trigger Mod
    {ACPI_DMT_FLAG1,    ACPI_GTDT_FLAG_OFFSET (VirtualTimerFlags,0),"Polarity",
    ACPI_DMT_NEW_LINE,
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (NonSecurePl2Interrupt),   "Non-Secure
    {ACPI_DMT_UINT32,   ACPI_GTDT_OFFSET (NonSecurePl2Flags),       "NSPL2 Flags
    {ACPI_DMT_FLAG0,    ACPI_GTDT_FLAG_OFFSET (NonSecurePl2Flags,0),"Trigger Mod
    {ACPI_DMT_FLAG1,    ACPI_GTDT_FLAG_OFFSET (NonSecurePl2Flags,0),"Polarity",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * HEST - Hardware Error Source table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoHest[] =
{
    {ACPI_DMT_UINT32,   ACPI_HEST_OFFSET (ErrorSourceCount),        "Error Sourc
    ACPI_DMT_TERMINATOR
};

/* Common HEST structures for subtables */

#define ACPI_DM_HEST_HEADER \
    {ACPI_DMT_HEST,     ACPI_HEST0_OFFSET (Header.Type),            "Subtable Ty
    {ACPI_DMT_UINT16,   ACPI_HEST0_OFFSET (Header.SourceId),        "Source Id",

#define ACPI_DM_HEST_AER \
    {ACPI_DMT_UINT16,   ACPI_HEST6_OFFSET (Aer.Reserved1),              "Reserve
    {ACPI_DMT_UINT8,    ACPI_HEST6_OFFSET (Aer.Flags),                  "Flags (
    {ACPI_DMT_FLAG0,    ACPI_HEST6_FLAG_OFFSET (Aer.Flags,0),           "Firmwar
    {ACPI_DMT_UINT8,    ACPI_HEST6_OFFSET (Aer.Enabled),                "Enabled
    {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.RecordsToPreallocate),   "Records
    {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.MaxSectionsPerRecord),   "Max Sec
    {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.Bus),                    "Bus", 0
```

```
1050     {ACPI_DMT_UINT16,   ACPI_HEST6_OFFSET (Aer.Device),                 "Device"
1051     {ACPI_DMT_UINT16,   ACPI_HEST6_OFFSET (Aer.Function),               "Functio
1052     {ACPI_DMT_UINT16,   ACPI_HEST6_OFFSET (Aer.DeviceControl),          "DeviceC
1053     {ACPI_DMT_UINT16,   ACPI_HEST6_OFFSET (Aer.Reserved2),              "Reserve
1054     {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.UncorrectableMask),      "Uncorre
1055     {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.UncorrectableSeverity),  "Uncorre
1056     {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.CorrectableMask),        "Correct
1057     {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (Aer.AdvancedCapabilities),   "Advance


1060 /* HEST Subtables */

1062 /* 0: IA32 Machine Check Exception */

1064 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest0[] =
1065 {
1066     ACPI_DM_HEST_HEADER,
1067     {ACPI_DMT_UINT16,   ACPI_HEST0_OFFSET (Reserved1),              "Reserved1",
1068     {ACPI_DMT_UINT8,    ACPI_HEST0_OFFSET (Flags),                  "Flags (deco
1069     {ACPI_DMT_FLAG0,    ACPI_HEST0_FLAG_OFFSET (Flags,0),           "Firmware Fi

1071     {ACPI_DMT_UINT8,    ACPI_HEST0_OFFSET (Enabled),                "Enabled", 0
1072     {ACPI_DMT_UINT32,   ACPI_HEST0_OFFSET (RecordsToPreallocate),   "Records To
1073     {ACPI_DMT_UINT32,   ACPI_HEST0_OFFSET (MaxSectionsPerRecord),   "Max Section
1074     {ACPI_DMT_UINT64,   ACPI_HEST0_OFFSET (GlobalCapabilityData),   "Global Capa
1075     {ACPI_DMT_UINT64,   ACPI_HEST0_OFFSET (GlobalControlData),      "Global Cont
1076     {ACPI_DMT_UINT8,    ACPI_HEST0_OFFSET (NumHardwareBanks),       "Num Hardwar
1077     {ACPI_DMT_UINT56,   ACPI_HEST0_OFFSET (Reserved3[0]),           "Reserved2",
1078     ACPI_DMT_TERMINATOR
1079 };

1081 /* 1: IA32 Corrected Machine Check */

1083 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest1[] =
1084 {
1085     ACPI_DM_HEST_HEADER,
1086     {ACPI_DMT_UINT16,   ACPI_HEST1_OFFSET (Reserved1),              "Reserved1",
1087     {ACPI_DMT_UINT8,    ACPI_HEST1_OFFSET (Flags),                  "Flags (deco
1088     {ACPI_DMT_FLAG0,    ACPI_HEST1_FLAG_OFFSET (Flags,0),           "Firmware Fi

1090     {ACPI_DMT_UINT8,    ACPI_HEST1_OFFSET (Enabled),                "Enabled", 0
1091     {ACPI_DMT_UINT32,   ACPI_HEST1_OFFSET (RecordsToPreallocate),   "Records To
1092     {ACPI_DMT_UINT32,   ACPI_HEST1_OFFSET (MaxSectionsPerRecord),   "Max Section
1093     {ACPI_DMT_HESTNTFY, ACPI_HEST1_OFFSET (Notify),                 "Notify", 0}
1094     {ACPI_DMT_UINT8,    ACPI_HEST1_OFFSET (NumHardwareBanks),       "Num Hardwar
1095     {ACPI_DMT_UINT24,   ACPI_HEST1_OFFSET (Reserved2[0]),           "Reserved2",
1096     ACPI_DMT_TERMINATOR
1097 };

1099 /* 2: IA32 Non-Maskable Interrupt */

1101 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest2[] =
1102 {
1103     ACPI_DM_HEST_HEADER,
1104     {ACPI_DMT_UINT32,   ACPI_HEST2_OFFSET (Reserved),               "Reserved",
1105     {ACPI_DMT_UINT32,   ACPI_HEST2_OFFSET (RecordsToPreallocate),   "Records To
1106     {ACPI_DMT_UINT32,   ACPI_HEST2_OFFSET (MaxSectionsPerRecord),   "Max Section
1107     {ACPI_DMT_UINT32,   ACPI_HEST2_OFFSET (MaxRawDataLength),       "Max Raw Dat
1108     ACPI_DMT_TERMINATOR
1109 };

1111 /* 6: PCI Express Root Port AER */

1113 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest6[] =
1114 {
1115     ACPI_DM_HEST_HEADER,
```



```
1116     ACPI_DM_HEST_AER,
1117     {ACPI_DMT_UINT32,   ACPI_HEST6_OFFSET (RootErrorCommand),       "Root Error
1118     ACPI_DMT_TERMINATOR
1119 };

1121 /* 7: PCI Express AER (AER Endpoint) */

1123 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest7[] =
1124 {
1125     ACPI_DM_HEST_HEADER,
1126     ACPI_DM_HEST_AER,
1127     ACPI_DMT_TERMINATOR
1128 };

1130 /* 8: PCI Express/PCI-X Bridge AER */

1132 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest8[] =
1133 {
1134     ACPI_DM_HEST_HEADER,
1135     ACPI_DM_HEST_AER,
1136     {ACPI_DMT_UINT32,   ACPI_HEST8_OFFSET (UncorrectableMask2),     "2nd Uncorre
1137     {ACPI_DMT_UINT32,   ACPI_HEST8_OFFSET (UncorrectableSeverity2), "2nd Uncorre
1138     {ACPI_DMT_UINT32,   ACPI_HEST8_OFFSET (AdvancedCapabilities2),  "2nd Advance
1139     ACPI_DMT_TERMINATOR
1140 };

1142 /* 9: Generic Hardware Error Source */

1144 ACPI_DMTABLE_INFO           AcpiDmTableInfoHest9[] =
1145 {
1146     ACPI_DM_HEST_HEADER,
1147     {ACPI_DMT_UINT16,   ACPI_HEST9_OFFSET (RelatedSourceId),        "Related Sou
1148     {ACPI_DMT_UINT8,    ACPI_HEST9_OFFSET (Reserved),               "Reserved",
1149     {ACPI_DMT_UINT8,    ACPI_HEST9_OFFSET (Enabled),                "Enabled", 0
1150     {ACPI_DMT_UINT32,   ACPI_HEST9_OFFSET (RecordsToPreallocate),   "Records To
1151     {ACPI_DMT_UINT32,   ACPI_HEST9_OFFSET (MaxSectionsPerRecord),   "Max Section
1152     {ACPI_DMT_UINT32,   ACPI_HEST9_OFFSET (MaxRawDataLength),       "Max Raw Dat
1153     {ACPI_DMT_GAS,      ACPI_HEST9_OFFSET (ErrorStatusAddress),     "Error Statu
1154     {ACPI_DMT_HESTNTFY, ACPI_HEST9_OFFSET (Notify),                 "Notify", 0}
1155     {ACPI_DMT_UINT32,   ACPI_HEST9_OFFSET (ErrorBlockLength),       "Error Statu
1156     ACPI_DMT_TERMINATOR
1157 };

1159 ACPI_DMTABLE_INFO           AcpiDmTableInfoHestNotify[] =
1160 {
1161     {ACPI_DMT_HESTNTYP, ACPI_HESTN_OFFSET (Type),                   "Notify Type
1162     {ACPI_DMT_UINT8,    ACPI_HESTN_OFFSET (Length),                 "Notify Leng
1163     {ACPI_DMT_UINT16,   ACPI_HESTN_OFFSET (ConfigWriteEnable),      "Configurati
1164     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (PollInterval),           "PollInterva
1165     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (Vector),                 "Vector", 0}
1166     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (PollingThresholdValue),  "Polling Thr
1167     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (PollingThresholdWindow), "Polling Thr
1168     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (ErrorThresholdValue),    "Error Thres
1169     {ACPI_DMT_UINT32,   ACPI_HESTN_OFFSET (ErrorThresholdWindow),   "Error Thres
1170     ACPI_DMT_TERMINATOR
1171 };


1174 /*
1175  * IA32 Error Bank(s) - Follows the ACPI_HEST_IA_MACHINE_CHECK and
1176  * ACPI_HEST_IA_CORRECTED structures.
1177  */
1178 ACPI_DMTABLE_INFO           AcpiDmTableInfoHestBank[] =
1179 {
1180     {ACPI_DMT_UINT8,    ACPI_HESTB_OFFSET (BankNumber),             "Bank Number
1181     {ACPI_DMT_UINT8,    ACPI_HESTB_OFFSET (ClearStatusOnInit),      "Clear Statu
```

```
1182     {ACPI_DMT_UINT8,    ACPI_HESTB_OFFSET (StatusFormat),          "Status Form
1183     {ACPI_DMT_UINT8,    ACPI_HESTB_OFFSET (Reserved),              "Reserved",
1184     {ACPI_DMT_UINT32,   ACPI_HESTB_OFFSET (ControlRegister),       "Control Reg
1185     {ACPI_DMT_UINT64,   ACPI_HESTB_OFFSET (ControlData),           "Control Dat
1186     {ACPI_DMT_UINT32,   ACPI_HESTB_OFFSET (StatusRegister),        "Status Regi
1187     {ACPI_DMT_UINT32,   ACPI_HESTB_OFFSET (AddressRegister),       "Address Reg
1188     {ACPI_DMT_UINT32,   ACPI_HESTB_OFFSET (MiscRegister),          "Misc Regist
1189     ACPI_DMT_TERMINATOR
1190 };


1193 /*******************************************************************************
1194  *
1195  * HPET - High Precision Event Timer table
1196  *
1197  ******************************************************************************/

1199 ACPI_DMTABLE_INFO           AcpiDmTableInfoHpet[] =
1200 {
1201     {ACPI_DMT_UINT32,   ACPI_HPET_OFFSET (Id),                     "Hardware Bl
1202     {ACPI_DMT_GAS,      ACPI_HPET_OFFSET (Address),                "Timer Block
1203     {ACPI_DMT_UINT8,    ACPI_HPET_OFFSET (Sequence),               "Sequence Nu
1204     {ACPI_DMT_UINT16,   ACPI_HPET_OFFSET (MinimumTick),            "Minimum Clo
1205     {ACPI_DMT_UINT8,    ACPI_HPET_OFFSET (Flags),                  "Flags (deco
1206     {ACPI_DMT_FLAG0,    ACPI_HPET_FLAG_OFFSET (Flags,0),           "4K Page Pro
1207     {ACPI_DMT_FLAG1,    ACPI_HPET_FLAG_OFFSET (Flags,0),           "64K Page Pr
1208     ACPI_DMT_TERMINATOR
1209 };


1212 /*******************************************************************************
1213  *
1214  * IVRS - I/O Virtualization Reporting Structure
1215  *
1216  ******************************************************************************/

1218 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs[] =
1219 {
1220     {ACPI_DMT_UINT32,   ACPI_IVRS_OFFSET (Info),                   "Virtualizat
1221     {ACPI_DMT_UINT64,   ACPI_IVRS_OFFSET (Reserved),               "Reserved",
1222     ACPI_DMT_TERMINATOR
1223 };

1225 /* Common Subtable header (one per Subtable) */

1227 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrsHdr[] =
1228 {
1229     {ACPI_DMT_IVRS,     ACPI_IVRSH_OFFSET (Type),                  "Subtable Ty
1230     {ACPI_DMT_UINT8,    ACPI_IVRSH_OFFSET (Flags),                 "Flags", 0},
1231     {ACPI_DMT_UINT16,   ACPI_IVRSH_OFFSET (Length),                "Length", DT
1232     {ACPI_DMT_UINT16,   ACPI_IVRSH_OFFSET (DeviceId),              "DeviceId",
1233     ACPI_DMT_TERMINATOR
1234 };

1236 /* IVRS subtables */

1238 /* 0x10: I/O Virtualization Hardware Definition (IVHD) Block */

1240 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs0[] =
1241 {
1242     {ACPI_DMT_UINT16,   ACPI_IVRS0_OFFSET (CapabilityOffset),      "Capability
1243     {ACPI_DMT_UINT64,   ACPI_IVRS0_OFFSET (BaseAddress),           "Base Addres
1244     {ACPI_DMT_UINT16,   ACPI_IVRS0_OFFSET (PciSegmentGroup),       "PCI Segment
1245     {ACPI_DMT_UINT16,   ACPI_IVRS0_OFFSET (Info),                  "Virtualizat
1246     {ACPI_DMT_UINT32,   ACPI_IVRS0_OFFSET (Reserved),              "Reserved",
1247     ACPI_DMT_TERMINATOR
```



```
1248 };

1250 /* 0x20, 0x21, 0x22: I/O Virtualization Memory Definition (IVMD) Block */

1252 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs1[] =
1253 {
1254     {ACPI_DMT_UINT16,   ACPI_IVRS1_OFFSET (AuxData),               "Auxiliary D
1255     {ACPI_DMT_UINT64,   ACPI_IVRS1_OFFSET (Reserved),              "Reserved",
1256     {ACPI_DMT_UINT64,   ACPI_IVRS1_OFFSET (StartAddress),          "Start Addre
1257     {ACPI_DMT_UINT64,   ACPI_IVRS1_OFFSET (MemoryLength),          "Memory Leng
1258     ACPI_DMT_TERMINATOR
1259 };

1261 /* Device entry header for IVHD block */

1263 #define ACPI_DMT_IVRS_DE_HEADER \
1264     {ACPI_DMT_UINT8,    ACPI_IVRSD_OFFSET (Type),                  "Entry Type"
1265     {ACPI_DMT_UINT16,   ACPI_IVRSD_OFFSET (Id),                    "Device ID",
1266     {ACPI_DMT_UINT8,    ACPI_IVRSD_OFFSET (DataSetting),           "Data Settin

1268 /* 4-byte device entry */

1270 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs4[] =
1271 {
1272     ACPI_DMT_IVRS_DE_HEADER,
1273     {ACPI_DMT_EXIT,     0,                                         NULL, 0},
1274 };

1276 /* 8-byte device entry */

1278 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs8a[] =
1279 {
1280     ACPI_DMT_IVRS_DE_HEADER,
1281     {ACPI_DMT_UINT8,    ACPI_IVRS8A_OFFSET (Reserved1),            "Reserved",
1282     {ACPI_DMT_UINT16,   ACPI_IVRS8A_OFFSET (UsedId),               "Source Used
1283     {ACPI_DMT_UINT8,    ACPI_IVRS8A_OFFSET (Reserved2),            "Reserved",
1284     ACPI_DMT_TERMINATOR
1285 };

1287 /* 8-byte device entry */

1289 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs8b[] =
1290 {
1291     ACPI_DMT_IVRS_DE_HEADER,
1292     {ACPI_DMT_UINT32,   ACPI_IVRS8B_OFFSET (ExtendedData),         "Extended Da
1293     ACPI_DMT_TERMINATOR
1294 };

1296 /* 8-byte device entry */

1298 ACPI_DMTABLE_INFO           AcpiDmTableInfoIvrs8c[] =
1299 {
1300     ACPI_DMT_IVRS_DE_HEADER,
1301     {ACPI_DMT_UINT8,    ACPI_IVRS8C_OFFSET (Handle),               "Handle", 0}
1302     {ACPI_DMT_UINT16,   ACPI_IVRS8C_OFFSET (UsedId),               "Source Used
1303     {ACPI_DMT_UINT8,    ACPI_IVRS8C_OFFSET (Variety),              "Variety", 0
1304     ACPI_DMT_TERMINATOR
1305 };


1308 /*******************************************************************************
1309  *
1310  * MADT - Multiple APIC Description Table and subtables
1311  *
1312  ******************************************************************************/
```

```
1314 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt[] =
1315 {
1316     {ACPI_DMT_UINT32,   ACPI_MADT_OFFSET (Address),                 "Local Apic
1317     {ACPI_DMT_UINT32,   ACPI_MADT_OFFSET (Flags),                   "Flags (deco
1318     {ACPI_DMT_FLAG0,    ACPI_MADT_FLAG_OFFSET (Flags,0),            "PC-AT Compa
1319     ACPI_DMT_TERMINATOR
1320 };

1322 /* Common Subtable header (one per Subtable) */

1324 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadtHdr[] =
1325 {
1326     {ACPI_DMT_MADT,     ACPI_MADTH_OFFSET (Type),                   "Subtable Ty
1327     {ACPI_DMT_UINT8,    ACPI_MADTH_OFFSET (Length),                 "Length", DT
1328     ACPI_DMT_TERMINATOR
1329 };

1331 /* MADT Subtables */

1333 /* 0: processor APIC */

1335 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt0[] =
1336 {
1337     {ACPI_DMT_UINT8,    ACPI_MADT0_OFFSET (ProcessorId),            "Processor I
1338     {ACPI_DMT_UINT8,    ACPI_MADT0_OFFSET (Id),                     "Local Apic
1339     {ACPI_DMT_UINT32,   ACPI_MADT0_OFFSET (LapicFlags),             "Flags (deco
1340     {ACPI_DMT_FLAG0,    ACPI_MADT0_FLAG_OFFSET (LapicFlags,0),      "Processor E
1341     ACPI_DMT_TERMINATOR
1342 };

1344 /* 1: IO APIC */

1346 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt1[] =
1347 {
1348     {ACPI_DMT_UINT8,    ACPI_MADT1_OFFSET (Id),                     "I/O Apic ID
1349     {ACPI_DMT_UINT8,    ACPI_MADT1_OFFSET (Reserved),               "Reserved",
1350     {ACPI_DMT_UINT32,   ACPI_MADT1_OFFSET (Address),                "Address", 0
1351     {ACPI_DMT_UINT32,   ACPI_MADT1_OFFSET (GlobalIrqBase),          "Interrupt",
1352     ACPI_DMT_TERMINATOR
1353 };

1355 /* 2: Interrupt Override */

1357 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt2[] =
1358 {
1359     {ACPI_DMT_UINT8,    ACPI_MADT2_OFFSET (Bus),                    "Bus", 0},
1360     {ACPI_DMT_UINT8,    ACPI_MADT2_OFFSET (SourceIrq),              "Source", 0}
1361     {ACPI_DMT_UINT32,   ACPI_MADT2_OFFSET (GlobalIrq),              "Interrupt",
1362     {ACPI_DMT_UINT16,   ACPI_MADT2_OFFSET (IntiFlags),              "Flags (deco
1363     {ACPI_DMT_FLAGS0,   ACPI_MADT2_FLAG_OFFSET (IntiFlags,0),       "Polarity",
1364     {ACPI_DMT_FLAGS2,   ACPI_MADT2_FLAG_OFFSET (IntiFlags,0),       "Trigger Mod
1365     ACPI_DMT_TERMINATOR
1366 };

1368 /* 3: NMI Sources */

1370 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt3[] =
1371 {
1372     {ACPI_DMT_UINT16,   ACPI_MADT3_OFFSET (IntiFlags),              "Flags (deco
1373     {ACPI_DMT_FLAGS0,   ACPI_MADT3_FLAG_OFFSET (IntiFlags,0),       "Polarity",
1374     {ACPI_DMT_FLAGS2,   ACPI_MADT3_FLAG_OFFSET (IntiFlags,0),       "Trigger Mod
1375     {ACPI_DMT_UINT32,   ACPI_MADT3_OFFSET (GlobalIrq),              "Interrupt",
1376     ACPI_DMT_TERMINATOR
1377 };

1379 /* 4: Local APIC NMI */
```



```
1381 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt4[] =
1382 {
1383     {ACPI_DMT_UINT8,    ACPI_MADT4_OFFSET (ProcessorId),            "Processor I
1384     {ACPI_DMT_UINT16,   ACPI_MADT4_OFFSET (IntiFlags),              "Flags (deco
1385     {ACPI_DMT_FLAGS0,   ACPI_MADT4_FLAG_OFFSET (IntiFlags,0),       "Polarity",
1386     {ACPI_DMT_FLAGS2,   ACPI_MADT4_FLAG_OFFSET (IntiFlags,0),       "Trigger Mod
1387     {ACPI_DMT_UINT8,    ACPI_MADT4_OFFSET (Lint),                   "Interrupt I
1388     ACPI_DMT_TERMINATOR
1389 };

1391 /* 5: Address Override */

1393 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt5[] =
1394 {
1395     {ACPI_DMT_UINT16,   ACPI_MADT5_OFFSET (Reserved),               "Reserved",
1396     {ACPI_DMT_UINT64,   ACPI_MADT5_OFFSET (Address),                "APIC Addres
1397     ACPI_DMT_TERMINATOR
1398 };

1400 /* 6: I/O Sapic */

1402 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt6[] =
1403 {
1404     {ACPI_DMT_UINT8,    ACPI_MADT6_OFFSET (Id),                     "I/O Sapic I
1405     {ACPI_DMT_UINT8,    ACPI_MADT6_OFFSET (Reserved),               "Reserved",
1406     {ACPI_DMT_UINT32,   ACPI_MADT6_OFFSET (GlobalIrqBase),          "Interrupt B
1407     {ACPI_DMT_UINT64,   ACPI_MADT6_OFFSET (Address),                "Address", 0
1408     ACPI_DMT_TERMINATOR
1409 };

1411 /* 7: Local Sapic */

1413 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt7[] =
1414 {
1415     {ACPI_DMT_UINT8,    ACPI_MADT7_OFFSET (ProcessorId),            "Processor I
1416     {ACPI_DMT_UINT8,    ACPI_MADT7_OFFSET (Id),                     "Local Sapic
1417     {ACPI_DMT_UINT8,    ACPI_MADT7_OFFSET (Eid),                    "Local Sapic
1418     {ACPI_DMT_UINT24,   ACPI_MADT7_OFFSET (Reserved[0]),            "Reserved",
1419     {ACPI_DMT_UINT32,   ACPI_MADT7_OFFSET (LapicFlags),             "Flags (deco
1420     {ACPI_DMT_FLAG0,    ACPI_MADT7_FLAG_OFFSET (LapicFlags,0),      "Processor E
1421     {ACPI_DMT_UINT32,   ACPI_MADT7_OFFSET (Uid),                    "Processor U
1422     {ACPI_DMT_STRING,   ACPI_MADT7_OFFSET (UidString[0]),           "Processor U
1423     ACPI_DMT_TERMINATOR
1424 };

1426 /* 8: Platform Interrupt Source */

1428 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt8[] =
1429 {
1430     {ACPI_DMT_UINT16,   ACPI_MADT8_OFFSET (IntiFlags),              "Flags (deco
1431     {ACPI_DMT_FLAGS0,   ACPI_MADT8_FLAG_OFFSET (IntiFlags,0),       "Polarity",
1432     {ACPI_DMT_FLAGS2,   ACPI_MADT8_FLAG_OFFSET (IntiFlags,0),       "Trigger Mod
1433     {ACPI_DMT_UINT8,    ACPI_MADT8_OFFSET (Type),                   "InterruptTy
1434     {ACPI_DMT_UINT8,    ACPI_MADT8_OFFSET (Id),                     "Processor I
1435     {ACPI_DMT_UINT8,    ACPI_MADT8_OFFSET (Eid),                    "Processor E
1436     {ACPI_DMT_UINT8,    ACPI_MADT8_OFFSET (IoSapicVector),          "I/O Sapic V
1437     {ACPI_DMT_UINT32,   ACPI_MADT8_OFFSET (GlobalIrq),              "Interrupt",
1438     {ACPI_DMT_UINT32,   ACPI_MADT8_OFFSET (Flags),                  "Flags (deco
1439     {ACPI_DMT_FLAG0,    ACPI_MADT8_OFFSET (Flags),                  "CPEI Overri
1440     ACPI_DMT_TERMINATOR
1441 };

1443 /* 9: Processor Local X2_APIC (ACPI 4.0) */

1445 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt9[] =
```

```
1446 {
1447     {ACPI_DMT_UINT16,   ACPI_MADT9_OFFSET (Reserved),               "Reserved",
1448     {ACPI_DMT_UINT32,   ACPI_MADT9_OFFSET (LocalApicId),            "Processor x
1449     {ACPI_DMT_UINT32,   ACPI_MADT9_OFFSET (LapicFlags),             "Flags (deco
1450     {ACPI_DMT_FLAG0,    ACPI_MADT9_FLAG_OFFSET (LapicFlags,0),      "Processor E
1451     {ACPI_DMT_UINT32,   ACPI_MADT9_OFFSET (Uid),                    "Processor U
1452     ACPI_DMT_TERMINATOR
1453 };

1455 /* 10: Local X2_APIC NMI (ACPI 4.0) */

1457 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt10[] =
1458 {
1459     {ACPI_DMT_UINT16,   ACPI_MADT10_OFFSET (IntiFlags),             "Flags (deco
1460     {ACPI_DMT_FLAGS0,   ACPI_MADT10_FLAG_OFFSET (IntiFlags,0),      "Polarity",
1461     {ACPI_DMT_FLAGS2,   ACPI_MADT10_FLAG_OFFSET (IntiFlags,0),      "Trigger Mod
1462     {ACPI_DMT_UINT32,   ACPI_MADT10_OFFSET (Uid),                   "Processor U
1463     {ACPI_DMT_UINT8,    ACPI_MADT10_OFFSET (Lint),                  "Interrupt I
1464     {ACPI_DMT_UINT24,   ACPI_MADT10_OFFSET (Reserved[0]),           "Reserved",
1465     ACPI_DMT_TERMINATOR
1466 };

1468 /* 11: Generic Interrupt Controller (ACPI 5.0) */

1470 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt11[] =
1471 {
1472     {ACPI_DMT_UINT16,   ACPI_MADT11_OFFSET (Reserved),              "Reserved",
1473     {ACPI_DMT_UINT32,   ACPI_MADT11_OFFSET (GicId),                 "Local GIC H
1474     {ACPI_DMT_UINT32,   ACPI_MADT11_OFFSET (Uid),                   "Processor U
1475     {ACPI_DMT_UINT32,   ACPI_MADT11_OFFSET (Flags),                 "Flags (deco
1476     {ACPI_DMT_FLAG0,    ACPI_MADT11_FLAG_OFFSET (Flags,0),          "Processor E
1477     {ACPI_DMT_UINT32,   ACPI_MADT11_OFFSET (ParkingVersion),        "Parking Pro
1478     {ACPI_DMT_UINT32,   ACPI_MADT11_OFFSET (PerformanceInterrupt),  "Performance
1479     {ACPI_DMT_UINT64,   ACPI_MADT11_OFFSET (ParkedAddress),         "Parked Addr
1480     {ACPI_DMT_UINT64,   ACPI_MADT11_OFFSET (BaseAddress),           "Base Addres
1481     ACPI_DMT_TERMINATOR
1482 };

1484 /* 12: Generic Interrupt Distributor (ACPI 5.0) */

1486 ACPI_DMTABLE_INFO           AcpiDmTableInfoMadt12[] =
1487 {
1488     {ACPI_DMT_UINT16,   ACPI_MADT12_OFFSET (Reserved),              "Reserved",
1489     {ACPI_DMT_UINT32,   ACPI_MADT12_OFFSET (GicId),                 "Local GIC H
1490     {ACPI_DMT_UINT64,   ACPI_MADT12_OFFSET (BaseAddress),           "Base Addres
1491     {ACPI_DMT_UINT32,   ACPI_MADT12_OFFSET (GlobalIrqBase),         "Interrupt B
1492     {ACPI_DMT_UINT32,   ACPI_MADT12_OFFSET (Reserved2),             "Reserved",
1493    ACPI_DMT_TERMINATOR
1494 };


1497 /*******************************************************************************
1498  *
1499  * MCFG - PCI Memory Mapped Configuration table and Subtable
1500  *
1501  ******************************************************************************/

1503 ACPI_DMTABLE_INFO           AcpiDmTableInfoMcfg[] =
1504 {
1505     {ACPI_DMT_UINT64,   ACPI_MCFG_OFFSET (Reserved[0]),             "Reserved",
1506     ACPI_DMT_TERMINATOR
1507 };

1509 ACPI_DMTABLE_INFO           AcpiDmTableInfoMcfg0[] =
1510 {
1511     {ACPI_DMT_UINT64,   ACPI_MCFG0_OFFSET (Address),                "Base Addres
```



```
1512     {ACPI_DMT_UINT16,   ACPI_MCFG0_OFFSET (PciSegment),             "Segment Gro
1513     {ACPI_DMT_UINT8,    ACPI_MCFG0_OFFSET (StartBusNumber),         "Start Bus N
1514     {ACPI_DMT_UINT8,    ACPI_MCFG0_OFFSET (EndBusNumber),           "End Bus Num
1515     {ACPI_DMT_UINT32,   ACPI_MCFG0_OFFSET (Reserved),               "Reserved",
1516     ACPI_DMT_TERMINATOR
1517 };


1520 /*******************************************************************************
1521  *
1522  * MCHI - Management Controller Host Interface table
1523  *
1524  ******************************************************************************/

1526 ACPI_DMTABLE_INFO           AcpiDmTableInfoMchi[] =
1527 {
1528     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (InterfaceType),           "Interface T
1529     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (Protocol),                "Protocol",
1530     {ACPI_DMT_UINT64,   ACPI_MCHI_OFFSET (ProtocolData),            "Protocol Da
1531     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (InterruptType),           "Interrupt T
1532     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (Gpe),                     "Gpe", 0},
1533     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (PciDeviceFlag),           "Pci Device
1534     {ACPI_DMT_UINT32,   ACPI_MCHI_OFFSET (GlobalInterrupt),         "Global Inte
1535     {ACPI_DMT_GAS,      ACPI_MCHI_OFFSET (ControlRegister),         "Control Reg
1536     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (PciSegment),              "Pci Segment
1537     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (PciBus),                  "Pci Bus", 0
1538     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (PciDevice),               "Pci Device"
1539     {ACPI_DMT_UINT8,    ACPI_MCHI_OFFSET (PciFunction),             "Pci Functio
1540     ACPI_DMT_TERMINATOR
1541 };


1544 /*******************************************************************************
1545  *
1546  * MPST - Memory Power State Table
1547  *
1548  ******************************************************************************/

1550 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst[] =
1551 {
1552     {ACPI_DMT_UINT8,    ACPI_MPST_OFFSET (ChannelId),               "Channel ID"
1553     {ACPI_DMT_UINT24,   ACPI_MPST_OFFSET (Reserved1[0]),            "Reserved",
1554     {ACPI_DMT_UINT16,   ACPI_MPST_OFFSET (PowerNodeCount),          "Power Node
1555     {ACPI_DMT_UINT16,   ACPI_MPST_OFFSET (Reserved2),               "Reserved",
1556     ACPI_DMT_TERMINATOR
1557 };

1559 /* MPST subtables */

1561 /* 0: Memory Power Node Structure */

1563 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst0[] =
1564 {
1565     {ACPI_DMT_UINT8,    ACPI_MPST0_OFFSET (Flags),                  "Flags (deco
1566     {ACPI_DMT_FLAG0,    ACPI_MPST0_FLAG_OFFSET (Flags,0),           "Node Enable
1567     {ACPI_DMT_FLAG1,    ACPI_MPST0_FLAG_OFFSET (Flags,0),           "Power Manag
1568     {ACPI_DMT_FLAG2,    ACPI_MPST0_FLAG_OFFSET (Flags,0),           "Hot Plug Ca

1570     {ACPI_DMT_UINT8,    ACPI_MPST0_OFFSET (Reserved1),              "Reserved",
1571     {ACPI_DMT_UINT16,   ACPI_MPST0_OFFSET (NodeId),                 "Node ID", 0
1572     {ACPI_DMT_UINT32,   ACPI_MPST0_OFFSET (Length),                 "Length", 0}
1573     {ACPI_DMT_UINT64,   ACPI_MPST0_OFFSET (RangeAddress),           "Range Addre
1574     {ACPI_DMT_UINT64,   ACPI_MPST0_OFFSET (RangeLength),            "Range Lengt
1575     {ACPI_DMT_UINT32,   ACPI_MPST0_OFFSET (NumPowerStates),         "Num Power S
1576     {ACPI_DMT_UINT32,   ACPI_MPST0_OFFSET (NumPhysicalComponents),  "Num Physica
1577     ACPI_DMT_TERMINATOR
```

```
1578 };

1580 /* 0A: Sub-subtable - Memory Power State Structure (follows Memory Power Node ab

1582 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst0A[] =
1583 {
1584     {ACPI_DMT_UINT8,    ACPI_MPST0A_OFFSET (PowerState),           "Power State
1585     {ACPI_DMT_UINT8,    ACPI_MPST0A_OFFSET (InfoIndex),            "InfoIndex",
1586     ACPI_DMT_TERMINATOR
1587 };

1589 /* 0B: Sub-subtable - Physical Component ID Structure (follows Memory Power Stat

1591 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst0B[] =
1592 {
1593     {ACPI_DMT_UINT16,   ACPI_MPST0B_OFFSET (ComponentId),          "Component I
1594     ACPI_DMT_TERMINATOR
1595 };

1597 /* 01: Power Characteristics Count (follows all Power Node(s) above) */

1599 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst1[] =
1600 {
1601     {ACPI_DMT_UINT16,   ACPI_MPST1_OFFSET (CharacteristicsCount),   "Characteris
1602     {ACPI_DMT_UINT16,   ACPI_MPST1_OFFSET (Reserved),              "Reserved",
1603     ACPI_DMT_TERMINATOR
1604 };

1606 /* 02: Memory Power State Characteristics Structure */

1608 ACPI_DMTABLE_INFO           AcpiDmTableInfoMpst2[] =
1609 {
1610     {ACPI_DMT_UINT8,    ACPI_MPST2_OFFSET (StructureId),           "Structure I
1611     {ACPI_DMT_UINT8,    ACPI_MPST2_OFFSET (Flags),                 "Flags (deco
1612     {ACPI_DMT_FLAG0,    ACPI_MPST2_FLAG_OFFSET (Flags,0),          "Memory Pres
1613     {ACPI_DMT_FLAG1,    ACPI_MPST2_FLAG_OFFSET (Flags,0),          "Auto Entry"
1614     {ACPI_DMT_FLAG2,    ACPI_MPST2_FLAG_OFFSET (Flags,0),          "Auto Exit",

1616     {ACPI_DMT_UINT16,   ACPI_MPST2_OFFSET (Reserved1),             "Reserved",
1617     {ACPI_DMT_UINT32,   ACPI_MPST2_OFFSET (AveragePower),          "Average Pow
1618     {ACPI_DMT_UINT32,   ACPI_MPST2_OFFSET (PowerSaving),           "Power Savin
1619     {ACPI_DMT_UINT64,   ACPI_MPST2_OFFSET (ExitLatency),           "Exit Latenc
1620     {ACPI_DMT_UINT64,   ACPI_MPST2_OFFSET (Reserved2),             "Reserved",
1621     ACPI_DMT_TERMINATOR
1622 };


1625 /**************************************************************************
1626  *
1627  * MSCT - Maximum System Characteristics Table (ACPI 4.0)
1628  *
1629  **************************************************************************/

1631 ACPI_DMTABLE_INFO           AcpiDmTableInfoMsct[] =
1632 {
1633     {ACPI_DMT_UINT32,   ACPI_MSCT_OFFSET (ProximityOffset),        "Proximity O
1634     {ACPI_DMT_UINT32,   ACPI_MSCT_OFFSET (MaxProximityDomains),    "Max Proximi
1635     {ACPI_DMT_UINT32,   ACPI_MSCT_OFFSET (MaxClockDomains),        "Max Clock D
1636     {ACPI_DMT_UINT64,   ACPI_MSCT_OFFSET (MaxAddress),             "Max Physica
1637     ACPI_DMT_TERMINATOR
1638 };

1640 /* Subtable - Maximum Proximity Domain Information. Version 1 */

1642 ACPI_DMTABLE_INFO           AcpiDmTableInfoMsct0[] =
1643 {
```



```
1644     {ACPI_DMT_UINT8,    ACPI_MSCT0_OFFSET (Revision),              "Revision",
1645     {ACPI_DMT_UINT8,    ACPI_MSCT0_OFFSET (Length),                "Length", DT
1646     {ACPI_DMT_UINT32,   ACPI_MSCT0_OFFSET (RangeStart),            "Domain Rang
1647     {ACPI_DMT_UINT32,   ACPI_MSCT0_OFFSET (RangeEnd),              "Domain Rang
1648     {ACPI_DMT_UINT32,   ACPI_MSCT0_OFFSET (ProcessorCapacity),     "Processor C
1649     {ACPI_DMT_UINT64,   ACPI_MSCT0_OFFSET (MemoryCapacity),        "Memory Capa
1650     ACPI_DMT_TERMINATOR
1651 };


1654 /**************************************************************************
1655  *
1656  * MTMR - MID Timer Table
1657  *
1658  **************************************************************************/

1660 ACPI_DMTABLE_INFO           AcpiDmTableInfoMtmr[] =
1661 {
1662     ACPI_DMT_TERMINATOR
1663 };

1665 /* MTMR Subtables - MTMR Entry */

1667 ACPI_DMTABLE_INFO           AcpiDmTableInfoMtmr0[] =
1668 {
1669     {ACPI_DMT_GAS,      ACPI_MTMR0_OFFSET (PhysicalAddress),       "PhysicalAdd
1670     {ACPI_DMT_UINT32,   ACPI_MTMR0_OFFSET (Frequency),             "Frequency",
1671     {ACPI_DMT_UINT32,   ACPI_MTMR0_OFFSET (Irq),                   "IRQ", 0},
1672     ACPI_DMT_TERMINATOR
1673 };


1676 /**************************************************************************
1677  *
1678  * PCCT - Platform Communications Channel Table (ACPI 5.0)
1679  *
1680  **************************************************************************/

1682 ACPI_DMTABLE_INFO           AcpiDmTableInfoPcct[] =
1683 {
1684     {ACPI_DMT_UINT32,   ACPI_PCCT_OFFSET (Flags),                  "Flags (deco
1685     {ACPI_DMT_FLAG0,    ACPI_PCCT_FLAG_OFFSET (Flags,0),           "Doorbell",
1686     {ACPI_DMT_UINT64,   ACPI_PCCT_OFFSET (Reserved),               "Reserved",
1687     ACPI_DMT_TERMINATOR
1688 };

1690 /* PCCT subtables */

1692 ACPI_DMTABLE_INFO           AcpiDmTableInfoPcctHdr[] =
1693 {
1694     {ACPI_DMT_PCCT,     ACPI_PCCT0_OFFSET (Header.Type),           "Subtable Ty
1695     {ACPI_DMT_UINT8,    ACPI_PCCT0_OFFSET (Header.Length),         "Length", DT
1696     ACPI_DMT_TERMINATOR
1697 };

1699 /* 0: Generic Communications Subspace */

1701 ACPI_DMTABLE_INFO           AcpiDmTableInfoPcct0[] =
1702 {
1703     {ACPI_DMT_UINT48,   ACPI_PCCT0_OFFSET (Reserved[0]),           "Reserved",
1704     {ACPI_DMT_UINT64,   ACPI_PCCT0_OFFSET (BaseAddress),           "Base Addres
1705     {ACPI_DMT_UINT64,   ACPI_PCCT0_OFFSET (Length),                "Address Len
1706     {ACPI_DMT_GAS,      ACPI_PCCT0_OFFSET (DoorbellRegister),      "Doorbell Re
1707     {ACPI_DMT_UINT64,   ACPI_PCCT0_OFFSET (PreserveMask),          "Preserve Ma
1708     {ACPI_DMT_UINT64,   ACPI_PCCT0_OFFSET (WriteMask),             "Write Mask"
1709     {ACPI_DMT_UINT32,   ACPI_PCCT0_OFFSET (Latency),               "Command Lat
```

```
1710     {ACPI_DMT_UINT32,   ACPI_PCCT0_OFFSET (MaxAccessRate),          "Maximum Acc
1711     {ACPI_DMT_UINT16,   ACPI_PCCT0_OFFSET (MinTurnaroundTime),      "Minimum Tur
1712     ACPI_DMT_TERMINATOR
1713 };


1716 /*******************************************************************************
1717  *
1718  * PMTT - Platform Memory Topology Table
1719  *
1720  ******************************************************************************/

1722 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmtt[] =
1723 {
1724     {ACPI_DMT_UINT32,   ACPI_PMTT_OFFSET (Reserved),                "Reserved",
1725     ACPI_DMT_TERMINATOR
1726 };

1728 /* Common Subtable header (one per Subtable) */

1730 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmttHdr[] =
1731 {
1732     {ACPI_DMT_PMTT,     ACPI_PMTTH_OFFSET (Type),                   "Subtable Ty
1733     {ACPI_DMT_UINT8,    ACPI_PMTTH_OFFSET (Reserved1),              "Reserved",
1734     {ACPI_DMT_UINT16,   ACPI_PMTTH_OFFSET (Length),                 "Length", DT
1735     {ACPI_DMT_UINT16,   ACPI_PMTTH_OFFSET (Flags),                  "Flags (deco
1736     {ACPI_DMT_FLAG0,    ACPI_PMTTH_FLAG_OFFSET (Flags,0),           "Top-level D
1737     {ACPI_DMT_FLAG1,    ACPI_PMTTH_FLAG_OFFSET (Flags,0),           "Physical El
1738     {ACPI_DMT_FLAGS2,   ACPI_PMTTH_FLAG_OFFSET (Flags,0),           "Memory Type
1739     {ACPI_DMT_UINT16,   ACPI_PMTTH_OFFSET (Reserved2),              "Reserved",
1740     ACPI_DMT_TERMINATOR
1741 };

1743 /* PMTT Subtables */

1745 /* 0: Socket */

1747 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmtt0[] =
1748 {
1749     {ACPI_DMT_UINT16,   ACPI_PMTT0_OFFSET (SocketId),               "Socket ID",
1750     {ACPI_DMT_UINT16,   ACPI_PMTT0_OFFSET (Reserved),               "Reserved",
1751     ACPI_DMT_TERMINATOR
1752 };

1754 /* 1: Memory Controller */

1756 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmtt1[] =
1757 {
1758     {ACPI_DMT_UINT32,   ACPI_PMTT1_OFFSET (ReadLatency),            "Read Latenc
1759     {ACPI_DMT_UINT32,   ACPI_PMTT1_OFFSET (WriteLatency),           "Write Laten
1760     {ACPI_DMT_UINT32,   ACPI_PMTT1_OFFSET (ReadBandwidth),          "Read Bandwi
1761     {ACPI_DMT_UINT32,   ACPI_PMTT1_OFFSET (WriteBandwidth),         "Write Bandw
1762     {ACPI_DMT_UINT16,   ACPI_PMTT1_OFFSET (AccessWidth),            "Access Widt
1763     {ACPI_DMT_UINT16,   ACPI_PMTT1_OFFSET (Alignment),              "Alignment",
1764     {ACPI_DMT_UINT16,   ACPI_PMTT1_OFFSET (Reserved),               "Reserved",
1765     {ACPI_DMT_UINT16,   ACPI_PMTT1_OFFSET (DomainCount),            "Domain Coun
1766     ACPI_DMT_TERMINATOR
1767 };

1769 /* 1a: Proximity Domain */

1771 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmtt1a[] =
1772 {
1773     {ACPI_DMT_UINT32,   ACPI_PMTT1A_OFFSET (ProximityDomain),       "Proximity D
1774     ACPI_DMT_TERMINATOR
1775 };
```



```
1777 /* 2: Physical Component */

1779 ACPI_DMTABLE_INFO           AcpiDmTableInfoPmtt2[] =
1780 {
1781     {ACPI_DMT_UINT16,   ACPI_PMTT2_OFFSET (ComponentId),            "Component I
1782     {ACPI_DMT_UINT16,   ACPI_PMTT2_OFFSET (Reserved),               "Reserved",
1783     {ACPI_DMT_UINT32,   ACPI_PMTT2_OFFSET (MemorySize),             "Memory Size
1784     {ACPI_DMT_UINT32,   ACPI_PMTT2_OFFSET (BiosHandle),             "Bios Handle
1785     ACPI_DMT_TERMINATOR
1786 };


1789 /*******************************************************************************
1790  *
1791  * S3PT - S3 Performance Table
1792  *
1793  ******************************************************************************/

1795 ACPI_DMTABLE_INFO           AcpiDmTableInfoS3pt[] =
1796 {
1797     {ACPI_DMT_SIG,      ACPI_S3PT_OFFSET (Signature[0]),             "Signature",
1798     {ACPI_DMT_UINT32,   ACPI_S3PT_OFFSET (Length),                  "Length", DT
1799     ACPI_DMT_TERMINATOR
1800 };

1802 /* S3PT subtable header */

1804 ACPI_DMTABLE_INFO           AcpiDmTableInfoS3ptHdr[] =
1805 {
1806     {ACPI_DMT_UINT16,   ACPI_S3PTH_OFFSET (Type),                   "Type", 0},
1807     {ACPI_DMT_UINT8,    ACPI_S3PTH_OFFSET (Length),                 "Length", DT
1808     {ACPI_DMT_UINT8,    ACPI_S3PTH_OFFSET (Revision),               "Revision",
1809     ACPI_DMT_TERMINATOR
1810 };

1812 /* 0: Basic S3 Resume Performance Record */

1814 ACPI_DMTABLE_INFO           AcpiDmTableInfoS3pt0[] =
1815 {
1816     {ACPI_DMT_UINT32,   ACPI_S3PT0_OFFSET (ResumeCount),            "Resume Coun
1817     {ACPI_DMT_UINT64,   ACPI_S3PT0_OFFSET (FullResume),             "Full Resume
1818     {ACPI_DMT_UINT64,   ACPI_S3PT0_OFFSET (AverageResume),          "Average Res
1819     ACPI_DMT_TERMINATOR
1820 };

1822 /* 1: Basic S3 Suspend Performance Record */

1824 ACPI_DMTABLE_INFO           AcpiDmTableInfoS3pt1[] =
1825 {
1826     {ACPI_DMT_UINT64,   ACPI_S3PT1_OFFSET (SuspendStart),           "Suspend Sta
1827     {ACPI_DMT_UINT64,   ACPI_S3PT1_OFFSET (SuspendEnd),             "Suspend End
1828     ACPI_DMT_TERMINATOR
1829 };


1832 /*******************************************************************************
1833  *
1834  * SBST - Smart Battery Specification Table
1835  *
1836  ******************************************************************************/

1838 ACPI_DMTABLE_INFO           AcpiDmTableInfoSbst[] =
1839 {
1840     {ACPI_DMT_UINT32,   ACPI_SBST_OFFSET (WarningLevel),            "Warning Lev
1841     {ACPI_DMT_UINT32,   ACPI_SBST_OFFSET (LowLevel),                "Low Level",
```

```
    {ACPI_DMT_UINT32,   ACPI_SBST_OFFSET (CriticalLevel),          "Critical Le
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * SLIC - Software Licensing Description Table. There is no common table, just
 * the standard ACPI header and then subtables.
 *
 ******************************************************************************/

/* Common Subtable header (one per Subtable) */

ACPI_DMTABLE_INFO           AcpiDmTableInfoSlicHdr[] =
{
    {ACPI_DMT_SLIC,     ACPI_SLICH_OFFSET (Type),                  "Subtable Ty
    {ACPI_DMT_UINT32,   ACPI_SLICH_OFFSET (Length),                "Length", DT
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoSlic0[] =
{
    {ACPI_DMT_UINT8,    ACPI_SLIC0_OFFSET (KeyType),               "Key Type",
    {ACPI_DMT_UINT8,    ACPI_SLIC0_OFFSET (Version),               "Version", 0
    {ACPI_DMT_UINT16,   ACPI_SLIC0_OFFSET (Reserved),              "Reserved",
    {ACPI_DMT_UINT32,   ACPI_SLIC0_OFFSET (Algorithm),             "Algorithm",
    {ACPI_DMT_NAME4,    ACPI_SLIC0_OFFSET (Magic),                 "Magic", 0},
    {ACPI_DMT_UINT32,   ACPI_SLIC0_OFFSET (BitLength),             "BitLength",
    {ACPI_DMT_UINT32,   ACPI_SLIC0_OFFSET (Exponent),              "Exponent",
    {ACPI_DMT_BUF128,   ACPI_SLIC0_OFFSET (Modulus[0]),            "Modulus", 0
    ACPI_DMT_TERMINATOR
};

ACPI_DMTABLE_INFO           AcpiDmTableInfoSlic1[] =
{
    {ACPI_DMT_UINT32,   ACPI_SLIC1_OFFSET (Version),               "Version", 0
    {ACPI_DMT_NAME6,    ACPI_SLIC1_OFFSET (OemId[0]),              "Oem ID", 0},
    {ACPI_DMT_NAME8,    ACPI_SLIC1_OFFSET (OemTableId[0]),         "Oem Table I
    {ACPI_DMT_NAME8,    ACPI_SLIC1_OFFSET (WindowsFlag[0]),        "Windows Fla
    {ACPI_DMT_UINT32,   ACPI_SLIC1_OFFSET (SlicVersion),           "SLIC Versio
    {ACPI_DMT_BUF16,    ACPI_SLIC1_OFFSET (Reserved[0]),           "Reserved",
    {ACPI_DMT_BUF128,   ACPI_SLIC1_OFFSET (Signature[0]),          "Signature",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * SLIT - System Locality Information Table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoSlit[] =
{
    {ACPI_DMT_UINT64,   ACPI_SLIT_OFFSET (LocalityCount),          "Localities",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * SPCR - Serial Port Console Redirection table
 *
 ******************************************************************************/
```



```
ACPI_DMTABLE_INFO           AcpiDmTableInfoSpcr[] =
{
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (InterfaceType),          "Interface T
    {ACPI_DMT_UINT24,   ACPI_SPCR_OFFSET (Reserved[0]),            "Reserved",
    {ACPI_DMT_GAS,      ACPI_SPCR_OFFSET (SerialPort),             "Serial Port
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (InterruptType),          "Interrupt T
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (PcInterrupt),            "PCAT-compat
    {ACPI_DMT_UINT32,   ACPI_SPCR_OFFSET (Interrupt),              "Interrupt",
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (BaudRate),               "Baud Rate",
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (Parity),                 "Parity", 0}
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (StopBits),               "Stop Bits",
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (FlowControl),            "Flow Contro
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (TerminalType),           "Terminal Ty
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (Reserved2),              "Reserved",
    {ACPI_DMT_UINT16,   ACPI_SPCR_OFFSET (PciDeviceId),            "PCI Device
    {ACPI_DMT_UINT16,   ACPI_SPCR_OFFSET (PciVendorId),            "PCI Vendor
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (PciBus),                 "PCI Bus", 0
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (PciDevice),              "PCI Device"
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (PciFunction),            "PCI Functio
    {ACPI_DMT_UINT32,   ACPI_SPCR_OFFSET (PciFlags),               "PCI Flags",
    {ACPI_DMT_UINT8,    ACPI_SPCR_OFFSET (PciSegment),             "PCI Segment
    {ACPI_DMT_UINT32,   ACPI_SPCR_OFFSET (Reserved2),              "Reserved",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * SPMI - Server Platform Management Interface table
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoSpmi[] =
{
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (InterfaceType),          "Interface T
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (Reserved),               "Reserved",
    {ACPI_DMT_UINT16,   ACPI_SPMI_OFFSET (SpecRevision),           "IPMI Spec V
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (InterruptType),          "Interrupt T
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (GpeNumber),              "GPE Number"
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (Reserved1),              "Reserved",
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (PciDeviceFlag),          "PCI Device
    {ACPI_DMT_UINT32,   ACPI_SPMI_OFFSET (Interrupt),              "Interrupt",
    {ACPI_DMT_GAS,      ACPI_SPMI_OFFSET (IpmiRegister),           "IPMI Regist
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (PciSegment),             "PCI Segment
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (PciBus),                 "PCI Bus", 0
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (PciDevice),              "PCI Device"
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (PciFunction),            "PCI Functio
    {ACPI_DMT_UINT8,    ACPI_SPMI_OFFSET (Reserved2),              "Reserved",
    ACPI_DMT_TERMINATOR
};


/*******************************************************************************
 *
 * SRAT - System Resource Affinity Table and Subtables
 *
 ******************************************************************************/

ACPI_DMTABLE_INFO           AcpiDmTableInfoSrat[] =
{
    {ACPI_DMT_UINT32,   ACPI_SRAT_OFFSET (TableRevision),          "Table Revis
    {ACPI_DMT_UINT64,   ACPI_SRAT_OFFSET (Reserved),               "Reserved",
    ACPI_DMT_TERMINATOR
};

/* Common Subtable header (one per Subtable) */
```

```
1975 ACPI_DMTABLE_INFO           AcpiDmTableInfoSratHdr[] =
1976 {
1977     {ACPI_DMT_SRAT,     ACPI_SRATH_OFFSET (Type),                   "Subtable Ty
1978     {ACPI_DMT_UINT8,    ACPI_SRATH_OFFSET (Length),                 "Length", DT
1979     ACPI_DMT_TERMINATOR
1980 };

1982 /* SRAT Subtables */

1984 /* 0: Processor Local APIC/SAPIC Affinity */

1986 ACPI_DMTABLE_INFO           AcpiDmTableInfoSrat0[] =
1987 {
1988     {ACPI_DMT_UINT8,    ACPI_SRAT0_OFFSET (ProximityDomainLo),      "Proximity D
1989     {ACPI_DMT_UINT8,    ACPI_SRAT0_OFFSET (ApicId),                 "Apic ID", 0
1990     {ACPI_DMT_UINT32,   ACPI_SRAT0_OFFSET (Flags),                  "Flags (deco
1991     {ACPI_DMT_FLAG0,    ACPI_SRAT0_FLAG_OFFSET (Flags,0),           "Enabled", 0
1992     {ACPI_DMT_UINT8,    ACPI_SRAT0_OFFSET (LocalSapicEid),          "Local Sapic
1993     {ACPI_DMT_UINT24,   ACPI_SRAT0_OFFSET (ProximityDomainHi[0]),   "Proximity D
1994     {ACPI_DMT_UINT32,   ACPI_SRAT0_OFFSET (Reserved),               "Reserved",
1995     ACPI_DMT_TERMINATOR
1996 };

1998 /* 1: Memory Affinity */

2000 ACPI_DMTABLE_INFO           AcpiDmTableInfoSrat1[] =
2001 {
2002     {ACPI_DMT_UINT32,   ACPI_SRAT1_OFFSET (ProximityDomain),        "Proximity D
2003     {ACPI_DMT_UINT16,   ACPI_SRAT1_OFFSET (Reserved),               "Reserved1",
2004     {ACPI_DMT_UINT64,   ACPI_SRAT1_OFFSET (BaseAddress),            "Base Addres
2005     {ACPI_DMT_UINT64,   ACPI_SRAT1_OFFSET (Length),                 "Address Len
2006     {ACPI_DMT_UINT32,   ACPI_SRAT1_OFFSET (Reserved1),              "Reserved2",
2007     {ACPI_DMT_UINT32,   ACPI_SRAT1_OFFSET (Flags),                  "Flags (deco
2008     {ACPI_DMT_FLAG0,    ACPI_SRAT1_FLAG_OFFSET (Flags,0),           "Enabled", 0
2009     {ACPI_DMT_FLAG1,    ACPI_SRAT1_FLAG_OFFSET (Flags,0),           "Hot Pluggab
2010     {ACPI_DMT_FLAG2,    ACPI_SRAT1_FLAG_OFFSET (Flags,0),           "Non-Volatil
2011     {ACPI_DMT_UINT64,   ACPI_SRAT1_OFFSET (Reserved2),              "Reserved3",
2012     ACPI_DMT_TERMINATOR
2013 };

2015 /* 2: Processor Local X2_APIC Affinity (ACPI 4.0) */

2017 ACPI_DMTABLE_INFO           AcpiDmTableInfoSrat2[] =
2018 {
2019     {ACPI_DMT_UINT16,   ACPI_SRAT2_OFFSET (Reserved),               "Reserved1",
2020     {ACPI_DMT_UINT32,   ACPI_SRAT2_OFFSET (ProximityDomain),        "Proximity D
2021     {ACPI_DMT_UINT32,   ACPI_SRAT2_OFFSET (ApicId),                 "Apic ID", 0
2022     {ACPI_DMT_UINT32,   ACPI_SRAT2_OFFSET (Flags),                  "Flags (deco
2023     {ACPI_DMT_FLAG0,    ACPI_SRAT2_FLAG_OFFSET (Flags,0),           "Enabled", 0
2024     {ACPI_DMT_UINT32,   ACPI_SRAT2_OFFSET (ClockDomain),            "Clock Domai
2025     {ACPI_DMT_UINT32,   ACPI_SRAT2_OFFSET (Reserved2),              "Reserved2",
2026     ACPI_DMT_TERMINATOR
2027 };


2030 /*******************************************************************************
2031  *
2032  * TCPA - Trusted Computing Platform Alliance table
2033  *
2034  ******************************************************************************/

2036 ACPI_DMTABLE_INFO           AcpiDmTableInfoTcpa[] =
2037 {
2038     {ACPI_DMT_UINT16,   ACPI_TCPA_OFFSET (Reserved),                "Reserved",
2039     {ACPI_DMT_UINT32,   ACPI_TCPA_OFFSET (MaxLogLength),            "Max Event L
```



```
2040     {ACPI_DMT_UINT64,   ACPI_TCPA_OFFSET (LogAddress),              "Event Log A
2041     ACPI_DMT_TERMINATOR
2042 };


2045 /*******************************************************************************
2046  *
2047  * TPM2 - Trusted Platform Module (TPM) 2.0 Hardware Interface Table
2048  *
2049  ******************************************************************************/

2051 ACPI_DMTABLE_INFO           AcpiDmTableInfoTpm2[] =
2052 {
2053     {ACPI_DMT_UINT32,   ACPI_TPM2_OFFSET (Flags),                   "Flags", 0},
2054     {ACPI_DMT_UINT64,   ACPI_TPM2_OFFSET (ControlAddress),          "Control Add
2055     {ACPI_DMT_UINT32,   ACPI_TPM2_OFFSET (StartMethod),             "Start Metho
2056     ACPI_DMT_TERMINATOR
2057 };


2060 /*******************************************************************************
2061  *
2062  * UEFI - UEFI Boot optimization Table
2063  *
2064  ******************************************************************************/

2066 ACPI_DMTABLE_INFO           AcpiDmTableInfoUefi[] =
2067 {
2068     {ACPI_DMT_UUID,     ACPI_UEFI_OFFSET (Identifier[0]),           "UUID Identi
2069     {ACPI_DMT_UINT16,   ACPI_UEFI_OFFSET (DataOffset),              "Data Offset
2070     ACPI_DMT_TERMINATOR
2071 };


2074 /*******************************************************************************
2075  *
2076  * VRTC - Virtual Real Time Clock Table
2077  *
2078  ******************************************************************************/

2080 ACPI_DMTABLE_INFO           AcpiDmTableInfoVrtc[] =
2081 {
2082     ACPI_DMT_TERMINATOR
2083 };

2085 /* VRTC Subtables - VRTC Entry */

2087 ACPI_DMTABLE_INFO           AcpiDmTableInfoVrtc0[] =
2088 {
2089     {ACPI_DMT_GAS,      ACPI_VRTC0_OFFSET (PhysicalAddress),        "PhysicalAdd
2090     {ACPI_DMT_UINT32,   ACPI_VRTC0_OFFSET (Irq),                    "IRQ", 0},
2091     ACPI_DMT_TERMINATOR
2092 };


2095 /*******************************************************************************
2096  *
2097  * WAET - Windows ACPI Emulated devices Table
2098  *
2099  ******************************************************************************/

2101 ACPI_DMTABLE_INFO           AcpiDmTableInfoWaet[] =
2102 {
2103     {ACPI_DMT_UINT32,   ACPI_WAET_OFFSET (Flags),                   "Flags (deco
2104     {ACPI_DMT_FLAG0,    ACPI_WAET_OFFSET (Flags),                   "RTC needs n
2105     {ACPI_DMT_FLAG1,    ACPI_WAET_OFFSET (Flags),                   "PM timer, o
```

```
2106     ACPI_DMT_TERMINATOR
2107 };


2110 /*******************************************************************************
2111  *
2112  * WDAT - Watchdog Action Table
2113  *
2114  ******************************************************************************/

2116 ACPI_DMTABLE_INFO           AcpiDmTableInfoWdat[] =
2117 {
2118     {ACPI_DMT_UINT32,   ACPI_WDAT_OFFSET (HeaderLength),            "Header Leng
2119     {ACPI_DMT_UINT16,   ACPI_WDAT_OFFSET (PciSegment),              "PCI Segment
2120     {ACPI_DMT_UINT8,    ACPI_WDAT_OFFSET (PciBus),                  "PCI Bus", 0
2121     {ACPI_DMT_UINT8,    ACPI_WDAT_OFFSET (PciDevice),               "PCI Device"
2122     {ACPI_DMT_UINT8,    ACPI_WDAT_OFFSET (PciFunction),             "PCI Functio
2123     {ACPI_DMT_UINT24,   ACPI_WDAT_OFFSET (Reserved[0]),             "Reserved",
2124     {ACPI_DMT_UINT32,   ACPI_WDAT_OFFSET (TimerPeriod),             "Timer Perio
2125     {ACPI_DMT_UINT32,   ACPI_WDAT_OFFSET (MaxCount),                "Max Count",
2126     {ACPI_DMT_UINT32,   ACPI_WDAT_OFFSET (MinCount),                "Min Count",
2127     {ACPI_DMT_UINT8,    ACPI_WDAT_OFFSET (Flags),                   "Flags (deco
2128     {ACPI_DMT_FLAG0,    ACPI_WDAT_OFFSET (Flags),                   "Enabled", 0
2129     {ACPI_DMT_FLAG7,    ACPI_WDAT_OFFSET (Flags),                   "Stopped Whe
2130     {ACPI_DMT_UINT24,   ACPI_WDAT_OFFSET (Reserved2[0]),            "Reserved",
2131     {ACPI_DMT_UINT32,   ACPI_WDAT_OFFSET (Entries),                 "Watchdog En
2132     ACPI_DMT_TERMINATOR
2133 };

2135 /* WDAT Subtables - Watchdog Instruction Entries */

2137 ACPI_DMTABLE_INFO           AcpiDmTableInfoWdat0[] =
2138 {
2139     {ACPI_DMT_UINT8,    ACPI_WDAT0_OFFSET (Action),                 "Watchdog Ac
2140     {ACPI_DMT_UINT8,    ACPI_WDAT0_OFFSET (Instruction),            "Instruction
2141     {ACPI_DMT_UINT16,   ACPI_WDAT0_OFFSET (Reserved),               "Reserved",
2142     {ACPI_DMT_GAS,      ACPI_WDAT0_OFFSET (RegisterRegion),         "Register Re
2143     {ACPI_DMT_UINT32,   ACPI_WDAT0_OFFSET (Value),                  "Value", 0},
2144     {ACPI_DMT_UINT32,   ACPI_WDAT0_OFFSET (Mask),                   "Register Ma
2145     ACPI_DMT_TERMINATOR
2146 };


2149 /*******************************************************************************
2150  *
2151  * WDDT - Watchdog Description Table
2152  *
2153  ******************************************************************************/

2155 ACPI_DMTABLE_INFO           AcpiDmTableInfoWddt[] =
2156 {
2157     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (SpecVersion),             "Specificati
2158     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (TableVersion),            "Table Versi
2159     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (PciVendorId),             "PCI Vendor
2160     {ACPI_DMT_GAS,      ACPI_WDDT_OFFSET (Address),                 "Timer Regis
2161     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (MaxCount),                "Max Count",
2162     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (MinCount),                "Min Count",
2163     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (Period),                  "Period", 0}
2164     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (Status),                  "Status (dec

2166     /* Status Flags byte 0 */

2168     {ACPI_DMT_FLAG0,    ACPI_WDDT_FLAG_OFFSET (Status,0),           "Available",
2169     {ACPI_DMT_FLAG1,    ACPI_WDDT_FLAG_OFFSET (Status,0),           "Active", 0}
2170     {ACPI_DMT_FLAG2,    ACPI_WDDT_FLAG_OFFSET (Status,0),           "OS Owns", 0
```



```
2172     /* Status Flags byte 1 */

2174     {ACPI_DMT_FLAG3,    ACPI_WDDT_FLAG_OFFSET (Status,1),           "User Reset"
2175     {ACPI_DMT_FLAG4,    ACPI_WDDT_FLAG_OFFSET (Status,1),           "Timeout Res
2176     {ACPI_DMT_FLAG5,    ACPI_WDDT_FLAG_OFFSET (Status,1),           "Power Fail
2177     {ACPI_DMT_FLAG6,    ACPI_WDDT_FLAG_OFFSET (Status,1),           "Unknown Res

2179     {ACPI_DMT_UINT16,   ACPI_WDDT_OFFSET (Capability),              "Capability

2181     /* Capability Flags byte 0 */

2183     {ACPI_DMT_FLAG0,    ACPI_WDDT_FLAG_OFFSET (Capability,0),       "Auto Reset"
2184     {ACPI_DMT_FLAG1,    ACPI_WDDT_FLAG_OFFSET (Capability,0),       "Timeout Ale
2185     ACPI_DMT_TERMINATOR
2186 };


2189 /*******************************************************************************
2190  *
2191  * WDRT - Watchdog Resource Table
2192  *
2193  ******************************************************************************/

2195 ACPI_DMTABLE_INFO           AcpiDmTableInfoWdrt[] =
2196 {
2197     {ACPI_DMT_GAS,      ACPI_WDRT_OFFSET (ControlRegister),         "Control Reg
2198     {ACPI_DMT_GAS,      ACPI_WDRT_OFFSET (CountRegister),           "Count Regis
2199     {ACPI_DMT_UINT16,   ACPI_WDRT_OFFSET (PciDeviceId),             "PCI Device
2200     {ACPI_DMT_UINT16,   ACPI_WDRT_OFFSET (PciVendorId),             "PCI Vendor
2201     {ACPI_DMT_UINT8,    ACPI_WDRT_OFFSET (PciBus),                  "PCI Bus", 0
2202     {ACPI_DMT_UINT8,    ACPI_WDRT_OFFSET (PciDevice),               "PCI Device"
2203     {ACPI_DMT_UINT8,    ACPI_WDRT_OFFSET (PciFunction),             "PCI Functio
2204     {ACPI_DMT_UINT8,    ACPI_WDRT_OFFSET (PciSegment),              "PCI Segment
2205     {ACPI_DMT_UINT16,   ACPI_WDRT_OFFSET (MaxCount),                "Max Count",
2206     {ACPI_DMT_UINT8,    ACPI_WDRT_OFFSET (Units),                   "Counter Uni
2207     ACPI_DMT_TERMINATOR
2208 };

2210 /*! [Begin] no source code translation */

2212 /*
2213  * Generic types (used in UEFI and custom tables)
2214  *
2215  * Examples:
2216  *
2217  *     Buffer : cc 04 ff bb
2218  *      UINT8 : 11
2219  *     UINT16 : 1122
2220  *     UINT24 : 112233
2221  *     UINT32 : 11223344
2222  *     UINT56 : 11223344556677
2223  *     UINT64 : 1122334455667788
2224  *
2225  *     String : "This is string"
2226  *    Unicode : "This string encoded to Unicode"
2227  *
2228  *       GUID : 11223344-5566-7788-99aa-bbccddeeff00
2229  * DevicePath : "\PciRoot(0)\Pci(0x1f,1)\Usb(0,0)"
2230  */

2232 #define ACPI_DM_GENERIC_ENTRY(FieldType, FieldName) \
2233     {{FieldType, 0, FieldName, 0}, ACPI_DMT_TERMINATOR}

2235 ACPI_DMTABLE_INFO           AcpiDmTableInfoGeneric[][2] =
2236 {
2237     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT8,      "UINT8"),
```

```
2238     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT16,     "UINT16"),
2239     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT24,     "UINT24"),
2240     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT32,     "UINT32"),
2241     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT40,     "UINT40"),
2242     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT48,     "UINT48"),
2243     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT56,     "UINT56"),
2244     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UINT64,     "UINT64"),
2245     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_STRING,     "String"),
2246     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UNICODE,    "Unicode"),
2247     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_BUFFER,     "Buffer"),
2248     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_UUID,       "GUID"),
2249     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_STRING,     "DevicePath"),
2250     ACPI_DM_GENERIC_ENTRY (ACPI_DMT_LABEL,      "Label"),
2251     {ACPI_DMT_TERMINATOR}
2252 };
2253 /*! [End] no source code translation !*/
```

```
**********************************************************
    7805 Thu Dec 26 13:48:25 2013
new/usr/src/common/acpica/common/getopt.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: getopt
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

/*
 * ACPICA getopt() implementation
 *
 * Option strings:
 *    "f"       - Option has no arguments
 *    "f:"      - Option requires an argument
 *    "f^"      - Option has optional single-char sub-options
 *    "f|"      - Option has required single-char sub-options
 */

#include <stdio.h>
#include <string.h>
#include "acpi.h"
#include "accommon.h"
#include "acapps.h"

#define ACPI_OPTION_ERROR(msg, badchar) \
    if (AcpiGbl_Opterr) {fprintf (stderr, "%s%c\n", msg, badchar);}


int                 AcpiGbl_Opterr = 1;
int                 AcpiGbl_Optind = 1;
int                 AcpiGbl_SubOptChar = 0;
char                *AcpiGbl_Optarg;

static int          CurrentCharPtr = 1;


/******************************************************************************
 *
 * FUNCTION:    AcpiGetoptArgument
 *
 * PARAMETERS:  argc, argv          - from main
 *
 * RETURN:      0 if an argument was found, -1 otherwise. Sets AcpiGbl_Optarg
 *              to point to the next argument.
 *
 * DESCRIPTION: Get the next argument. Used to obtain arguments for the
 *              two-character options after the original call to AcpiGetopt.
 *              Note: Either the argument starts at the next character after
 *              the option, or it is pointed to by the next argv entry.
 *              (After call to AcpiGetopt, we need to backup to the previous
 *              argv entry).
 *
 *****************************************************************************/

int
AcpiGetoptArgument (
    int                     argc,
    char                    **argv)
{
    AcpiGbl_Optind--;
    CurrentCharPtr++;

    if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
    {
        AcpiGbl_Optarg = &argv[AcpiGbl_Optind++][(int) (CurrentCharPtr+1)];
    }
    else if (++AcpiGbl_Optind >= argc)
    {
        ACPI_OPTION_ERROR ("Option requires an argument: -", 'v');

        CurrentCharPtr = 1;
        return (-1);
    }
    else
    {
        AcpiGbl_Optarg = argv[AcpiGbl_Optind++];
    }

    CurrentCharPtr = 1;
    return (0);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiGetopt
 *
 * PARAMETERS:  argc, argv          - from main
 *              opts                - options info list
 *
 * RETURN:      Option character or EOF
```

```
 *
 * DESCRIPTION: Get the next option
 *
 ******************************************************************************/

int
AcpiGetopt(
    int                     argc,
    char                    **argv,
    char                    *opts)
{
    int                     CurrentChar;
    char                    *OptsPtr;


    if (CurrentCharPtr == 1)
    {
        if (AcpiGbl_Optind >= argc ||
            argv[AcpiGbl_Optind][0] != '-' ||
            argv[AcpiGbl_Optind][1] == '\0')
        {
            return (EOF);
        }
        else if (strcmp (argv[AcpiGbl_Optind], "--") == 0)
        {
            AcpiGbl_Optind++;
            return (EOF);
        }
    }

    /* Get the option */

    CurrentChar = argv[AcpiGbl_Optind][CurrentCharPtr];

    /* Make sure that the option is legal */

    if (CurrentChar == ':' ||
       (OptsPtr = strchr (opts, CurrentChar)) == NULL)
    {
        ACPI_OPTION_ERROR ("Illegal option: -", CurrentChar);

        if (argv[AcpiGbl_Optind][++CurrentCharPtr] == '\0')
        {
            AcpiGbl_Optind++;
            CurrentCharPtr = 1;
        }

        return ('?');
    }

    /* Option requires an argument? */

    if (*++OptsPtr == ':')
    {
        if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
        {
            AcpiGbl_Optarg = &argv[AcpiGbl_Optind++][(int) (CurrentCharPtr+1)];
        }
        else if (++AcpiGbl_Optind >= argc)
        {
            ACPI_OPTION_ERROR ("Option requires an argument: -", CurrentChar);

            CurrentCharPtr = 1;
            return ('?');
        }
        else
```



```
        {
            AcpiGbl_Optarg = argv[AcpiGbl_Optind++];
        }

        CurrentCharPtr = 1;
    }

    /* Option has an optional argument? */

    else if (*OptsPtr == '+')
    {
        if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
        {
            AcpiGbl_Optarg = &argv[AcpiGbl_Optind++][(int) (CurrentCharPtr+1)];
        }
        else if (++AcpiGbl_Optind >= argc)
        {
            AcpiGbl_Optarg = NULL;
        }
        else
        {
            AcpiGbl_Optarg = argv[AcpiGbl_Optind++];
        }

        CurrentCharPtr = 1;
    }

    /* Option has optional single-char arguments? */

    else if (*OptsPtr == '^')
    {
        if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
        {
            AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
        }
        else
        {
            AcpiGbl_Optarg = "^";
        }

        AcpiGbl_SubOptChar = AcpiGbl_Optarg[0];
        AcpiGbl_Optind++;
        CurrentCharPtr = 1;
    }

    /* Option has a required single-char argument? */

    else if (*OptsPtr == '|')
    {
        if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
        {
            AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
        }
        else
        {
            ACPI_OPTION_ERROR ("Option requires a single-character suboption: -"

            CurrentCharPtr = 1;
            return ('?');
        }

        AcpiGbl_SubOptChar = AcpiGbl_Optarg[0];
        AcpiGbl_Optind++;
        CurrentCharPtr = 1;
    }
```

```
259     /* Option with no arguments */

261     else
262     {
263         if (argv[AcpiGbl_Optind][++CurrentCharPtr] == '\0')
264         {
265             CurrentCharPtr = 1;
266             AcpiGbl_Optind++;
267         }

269         AcpiGbl_Optarg = NULL;
270     }

272     return (CurrentChar);
273 }
```

```
**********************************************************
   15585 Thu Dec 26 13:48:25 2013
new/usr/src/common/acpica/compiler/aslanalyze.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslanalyze.c - Support functions for parse tree walks
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include <string.h>


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslanalyze")


/******************************************************************************
 *
 * FUNCTION:    AnIsInternalMethod
 *
 * PARAMETERS:  Op                  - Current op
 *
 * RETURN:      Boolean
```



```
 *
 * DESCRIPTION: Check for an internal control method.
 *
 *****************************************************************************/

BOOLEAN
AnIsInternalMethod (
    ACPI_PARSE_OBJECT       *Op)
{

    if ((!ACPI_STRCMP (Op->Asl.ExternalName, "\\_OSI")) ||
        (!ACPI_STRCMP (Op->Asl.ExternalName, "_OSI")))
    {
        return (TRUE);
    }

    return (FALSE);
}


/******************************************************************************
 *
 * FUNCTION:    AnGetInternalMethodReturnType
 *
 * PARAMETERS:  Op                  - Current op
 *
 * RETURN:      Btype
 *
 * DESCRIPTION: Get the return type of an internal method
 *
 *****************************************************************************/

UINT32
AnGetInternalMethodReturnType (
    ACPI_PARSE_OBJECT       *Op)
{

    if ((!ACPI_STRCMP (Op->Asl.ExternalName, "\\_OSI")) ||
        (!ACPI_STRCMP (Op->Asl.ExternalName, "_OSI")))
    {
        return (ACPI_BTYPE_STRING);
    }

    return (0);
}


/******************************************************************************
 *
 * FUNCTION:    AnCheckId
 *
 * PARAMETERS:  Op                  - Current parse op
 *              Type                - HID or CID
 *
 * RETURN:      None
 *
 * DESCRIPTION: Perform various checks on _HID and _CID strings. Only limited
 *              checks can be performed on _CID strings.
 *
 *****************************************************************************/

void
AnCheckId (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_NAME               Type)
{
```

```
127     UINT32                  i;
128     ACPI_SIZE               Length;


131     /* Only care about string versions of _HID/_CID (integers are legal) */

133     if (Op->Asl.ParseOpcode != PARSEOP_STRING_LITERAL)
134     {
135         return;
136     }

138     /* For both _HID and _CID, the string must be non-null */

140     Length = strlen (Op->Asl.Value.String);
141     if (!Length)
142     {
143         AslError (ASL_ERROR, ASL_MSG_NULL_STRING,
144             Op, NULL);
145         return;
146     }

148     /*
149      * One of the things we want to catch here is the use of a leading
150      * asterisk in the string -- an odd construct that certain platform
151      * manufacturers are fond of. Technically, a leading asterisk is OK
152      * for _CID, but a valid use of this has not been seen.
153      */
154     if (*Op->Asl.Value.String == '*')
155     {
156         AslError (ASL_ERROR, ASL_MSG_LEADING_ASTERISK,
157             Op, Op->Asl.Value.String);
158         return;
159     }

161     /* _CID strings are bus-specific, no more checks can be performed */

163     if (Type == ASL_TYPE_CID)
164     {
165         return;
166     }

168     /* For _HID, all characters must be alphanumeric */

170     for (i = 0; Op->Asl.Value.String[i]; i++)
171     {
172         if (!isalnum ((int) Op->Asl.Value.String[i]))
173         {
174             AslError (ASL_ERROR, ASL_MSG_ALPHANUMERIC_STRING,
175                 Op, Op->Asl.Value.String);
176             return;
177         }
178     }

180     /*
181      * _HID String must be one of these forms:
182      *
183      * "AAA####"    A is an uppercase letter and # is a hex digit
184      * "ACPI####"   # is a hex digit
185      * "NNNN####"   N is an uppercase letter or decimal digit (0-9)
186      *              # is a hex digit (ACPI 5.0)
187      */
188     if ((Length < 7) || (Length > 8))
189     {
190         AslError (ASL_ERROR, ASL_MSG_HID_LENGTH,
191             Op, Op->Asl.Value.String);
192         return;
```



```
193     }

195     /* _HID Length is valid (7 or 8), now check the prefix (first 3 or 4 chars)

197     if (Length == 7)
198     {
199         /* AAA####: Ensure the alphabetic prefix is all uppercase */

201         for (i = 0; i < 3; i++)
202         {
203             if (!isupper ((int) Op->Asl.Value.String[i]))
204             {
205                 AslError (ASL_ERROR, ASL_MSG_UPPER_CASE,
206                     Op, &Op->Asl.Value.String[i]);
207                 return;
208             }
209         }
210     }
211     else /* Length == 8 */
212     {
213         /*
214          * ACPI#### or NNNN####:
215          * Ensure the prefix contains only uppercase alpha or decimal digits
216          */
217         for (i = 0; i < 4; i++)
218         {
219             if (!isupper ((int) Op->Asl.Value.String[i]) &&
220                 !isdigit ((int) Op->Asl.Value.String[i]))
221             {
222                 AslError (ASL_ERROR, ASL_MSG_HID_PREFIX,
223                     Op, &Op->Asl.Value.String[i]);
224                 return;
225             }
226         }
227     }

229     /* Remaining characters (suffix) must be hex digits */

231     for (; i < Length; i++)
232     {
233         if (!isxdigit ((int) Op->Asl.Value.String[i]))
234         {
235          AslError (ASL_ERROR, ASL_MSG_HID_SUFFIX,
236             Op, &Op->Asl.Value.String[i]);
237             break;
238         }
239     }
240 }


243 /*******************************************************************************
244  *
245  * FUNCTION:    AnLastStatementIsReturn
246  *
247  * PARAMETERS:  Op                  - A method parse node
248  *
249  * RETURN:      TRUE if last statement is an ASL RETURN. False otherwise
250  *
251  * DESCRIPTION: Walk down the list of top level statements within a method
252  *              to find the last one. Check if that last statement is in
253  *              fact a RETURN statement.
254  *
255  ******************************************************************************/

257 BOOLEAN
258 AnLastStatementIsReturn (
```

```
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Next;


    /* Check if last statement is a return */

    Next = ASL_GET_CHILD_NODE (Op);
    while (Next)
    {
        if ((!Next->Asl.Next) &&
            (Next->Asl.ParseOpcode == PARSEOP_RETURN))
        {
            return (TRUE);
        }

        Next = ASL_GET_PEER_NODE (Next);
    }

    return (FALSE);
}


/*******************************************************************************
 *
 * FUNCTION:    AnCheckMethodReturnValue
 *
 * PARAMETERS:  Op                  - Parent
 *              OpInfo              - Parent info
 *              ArgOp               - Method invocation op
 *              RequiredBtypes      - What caller requires
 *              ThisNodeBtype       - What this node returns (if anything)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Check a method invocation for 1) A return value and if it does
 *              in fact return a value, 2) check the type of the return value.
 *
 ******************************************************************************/

void
AnCheckMethodReturnValue (
    ACPI_PARSE_OBJECT       *Op,
    const ACPI_OPCODE_INFO  *OpInfo,
    ACPI_PARSE_OBJECT       *ArgOp,
    UINT32                  RequiredBtypes,
    UINT32                  ThisNodeBtype)
{
    ACPI_PARSE_OBJECT       *OwningOp;
    ACPI_NAMESPACE_NODE     *Node;


    Node = ArgOp->Asl.Node;


    /* Examine the parent op of this method */

    OwningOp = Node->Op;
    if (OwningOp->Asl.CompileFlags & NODE_METHOD_NO_RETVAL)
    {
        /* Method NEVER returns a value */

        AslError (ASL_ERROR, ASL_MSG_NO_RETVAL, Op, Op->Asl.ExternalName);
    }
    else if (OwningOp->Asl.CompileFlags & NODE_METHOD_SOME_NO_RETVAL)
    {
```



```
        /* Method SOMETIMES returns a value, SOMETIMES not */

        AslError (ASL_WARNING, ASL_MSG_SOME_NO_RETVAL, Op, Op->Asl.ExternalName);
    }
    else if (!(ThisNodeBtype & RequiredBtypes))
    {
        /* Method returns a value, but the type is wrong */

        AnFormatBtype (StringBuffer, ThisNodeBtype);
        AnFormatBtype (StringBuffer2, RequiredBtypes);

        /*
         * The case where the method does not return any value at all
         * was already handled in the namespace cross reference
         * -- Only issue an error if the method in fact returns a value,
         * but it is of the wrong type
         */
        if (ThisNodeBtype != 0)
        {
            sprintf (MsgBuffer,
                "Method returns [%s], %s operator requires [%s]",
                StringBuffer, OpInfo->Name, StringBuffer2);

            AslError (ASL_ERROR, ASL_MSG_INVALID_TYPE, ArgOp, MsgBuffer);
        }
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AnIsResultUsed
 *
 * PARAMETERS:  Op                  - Parent op for the operator
 *
 * RETURN:      TRUE if result from this operation is actually consumed
 *
 * DESCRIPTION: Determine if the function result value from an operator is
 *              used.
 *
 ******************************************************************************/

BOOLEAN
AnIsResultUsed (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Parent;


    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_INCREMENT:
    case PARSEOP_DECREMENT:

        /* These are standalone operators, no return value */

        return (TRUE);

    default:

        break;
    }

    /* Examine parent to determine if the return value is used */

    Parent = Op->Asl.Parent;
```

```
391     switch (Parent->Asl.ParseOpcode)
392     {
393     /* If/While - check if the operator is the predicate */

395     case PARSEOP_IF:
396     case PARSEOP_WHILE:

398         /* First child is the predicate */

400         if (Parent->Asl.Child == Op)
401         {
402             return (TRUE);
403         }
404         return (FALSE);

406     /* Not used if one of these is the parent */

408     case PARSEOP_METHOD:
409     case PARSEOP_DEFINITIONBLOCK:
410     case PARSEOP_ELSE:

412         return (FALSE);

414     default:

416         /* Any other type of parent means that the result is used */

418         return (TRUE);
419     }
420 }


423 /*******************************************************************************
424  *
425  * FUNCTION:    ApCheckForGpeNameConflict
426  *
427  * PARAMETERS:  Op                  - Current parse op
428  *
429  * RETURN:      None
430  *
431  * DESCRIPTION: Check for a conflict between GPE names within this scope.
432  *              Conflict means two GPE names with the same GPE number, but
433  *              different types -- such as _L1C and _E1C.
434  *
435  ******************************************************************************/

437 void
438 ApCheckForGpeNameConflict (
439     ACPI_PARSE_OBJECT       *Op)
440 {
441     ACPI_PARSE_OBJECT       *NextOp;
442     UINT32                  GpeNumber;
443     char                    Name[ACPI_NAME_SIZE + 1];
444     char                    Target[ACPI_NAME_SIZE];


447     /* Need a null-terminated string version of NameSeg */

449     ACPI_MOVE_32_TO_32 (Name, &Op->Asl.NameSeg);
450     Name[ACPI_NAME_SIZE] = 0;

452     /*
453      * For a GPE method:
454      * 1st char must be underscore
455      * 2nd char must be L or E
456      * 3rd/4th chars must be a hex number
```



```
457      */
458     if ((Name[0] != '_') ||
459         ((Name[1] != 'L') && (Name[1] != 'E')))
460     {
461         return;
462     }

464     /* Verify 3rd/4th chars are a valid hex value */

466     GpeNumber = ACPI_STRTOUL (&Name[2], NULL, 16);
467     if (GpeNumber == ACPI_UINT32_MAX)
468     {
469         return;
470     }

472     /*
473      * We are now sure we have an _Lxx or _Exx.
474      * Create the target name that would cause collision (Flip E/L)
475      */
476     ACPI_MOVE_32_TO_32 (Target, Name);

478     /* Inject opposite letter ("L" versus "E") */

480     if (Name[1] == 'L')
481     {
482         Target[1] = 'E';
483     }
484     else /* Name[1] == 'E' */
485     {
486         Target[1] = 'L';
487     }

489     /* Search all peers (objects within this scope) for target match */

491     NextOp = Op->Asl.Next;
492     while (NextOp)
493     {
494         /*
495          * We mostly care about methods, but check Name() constructs also,
496          * even though they will get another error for not being a method.
497          * All GPE names must be defined as control methods.
498          */
499         if ((NextOp->Asl.ParseOpcode == PARSEOP_METHOD) ||
500             (NextOp->Asl.ParseOpcode == PARSEOP_NAME))
501         {
502             if (ACPI_COMPARE_NAME (Target, NextOp->Asl.NameSeg))
503             {
504                 /* Found both _Exy and _Lxy in the same scope, error */

506                 AslError (ASL_ERROR, ASL_MSG_GPE_NAME_CONFLICT, NextOp,
507                     Name);
508                 return;
509             }
510         }

512         NextOp = NextOp->Asl.Next;
513     }

515     /* OK, no conflict found */

517     return;
518 }


521 /*******************************************************************************
522  *
```

```
 * FUNCTION:    ApCheckRegMethod
 *
 * PARAMETERS:  Op                  - Current parse op
 *
 * RETURN:      None
 *
 * DESCRIPTION: Ensure that a _REG method has a corresponding Operation
 *              Region declaration within the same scope. Note: _REG is defined
 *              to have two arguments and must therefore be defined as a
 *              control method.
 *
 *****************************************************************************/

void
ApCheckRegMethod (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Next;
    ACPI_PARSE_OBJECT       *Parent;


    /* We are only interested in _REG methods */

    if (!ACPI_COMPARE_NAME (METHOD_NAME__REG, &Op->Asl.NameSeg))
    {
        return;
    }

    /* Get the start of the current scope */

    Parent = Op->Asl.Parent;
    Next = Parent->Asl.Child;

    /* Search entire scope for an operation region declaration */

    while (Next)
    {
        if (Next->Asl.ParseOpcode == PARSEOP_OPERATIONREGION)
        {
            return; /* Found region, OK */
        }

        Next = Next->Asl.Next;
    }

    /* No region found, issue warning */

    AslError (ASL_WARNING, ASL_MSG_NO_REGION, Op, NULL);
}
```

```
**********************************************************
   14222 Thu Dec 26 13:48:25 2013
new/usr/src/common/acpica/compiler/aslbtypes.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslbtypes - Support for bitfield types
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslbtypes")

/* Local prototypes */

static UINT32
AnMapEtypeToBtype (
    UINT32                  Etype);
```



```
/*******************************************************************************
 *
 * FUNCTION:    AnMapArgTypeToBtype
 *
 * PARAMETERS:  ArgType             - The ARGI required type(s) for this
 *                                    argument, from the opcode info table
 *
 * RETURN:      The corresponding Bit-encoded types
 *
 * DESCRIPTION: Convert an encoded ARGI required argument type code into a
 *              bitfield type code. Implements the implicit source conversion
 *              rules.
 *
 ******************************************************************************/

UINT32
AnMapArgTypeToBtype (
    UINT32                  ArgType)
{

    switch (ArgType)
    {

    /* Simple types */

    case ARGI_ANYTYPE:

        return (ACPI_BTYPE_OBJECTS_AND_REFS);

    case ARGI_PACKAGE:

        return (ACPI_BTYPE_PACKAGE);

    case ARGI_EVENT:

        return (ACPI_BTYPE_EVENT);

    case ARGI_MUTEX:

        return (ACPI_BTYPE_MUTEX);

    case ARGI_DDBHANDLE:
        /*
         * DDBHandleObject := SuperName
         * ACPI_BTYPE_REFERENCE: Index reference as parameter of Load/Unload
         */
        return (ACPI_BTYPE_DDB_HANDLE | ACPI_BTYPE_REFERENCE);

    /* Interchangeable types */
    /*
     * Source conversion rules:
     * Integer, String, and Buffer are all interchangeable
     */
    case ARGI_INTEGER:
    case ARGI_STRING:
    case ARGI_BUFFER:
    case ARGI_BUFFER_OR_STRING:
    case ARGI_COMPUTEDATA:

        return (ACPI_BTYPE_COMPUTE_DATA);

    /* References */

    case ARGI_INTEGER_REF:

        return (ACPI_BTYPE_INTEGER);
```

```
127     case ARGI_OBJECT_REF:

129         return (ACPI_BTYPE_ALL_OBJECTS);

131     case ARGI_DEVICE_REF:

133         return (ACPI_BTYPE_DEVICE_OBJECTS);

135     case ARGI_REFERENCE:

137         return (ACPI_BTYPE_REFERENCE);

139     case ARGI_TARGETREF:
140     case ARGI_FIXED_TARGET:
141     case ARGI_SIMPLE_TARGET:

143         return (ACPI_BTYPE_OBJECTS_AND_REFS);

145     /* Complex types */

147     case ARGI_DATAOBJECT:
148         /*
149          * Buffer, string, package or reference to a Op -
150          * Used only by SizeOf operator
151          */
152         return (ACPI_BTYPE_STRING | ACPI_BTYPE_BUFFER |
153             ACPI_BTYPE_PACKAGE | ACPI_BTYPE_REFERENCE);

155     case ARGI_COMPLEXOBJ:

157         /* Buffer, String, or package */

159         return (ACPI_BTYPE_STRING | ACPI_BTYPE_BUFFER | ACPI_BTYPE_PACKAGE);

161     case ARGI_REF_OR_STRING:

163         return (ACPI_BTYPE_STRING | ACPI_BTYPE_REFERENCE);

165     case ARGI_REGION_OR_BUFFER:

167         /* Used by Load() only. Allow buffers in addition to regions/fields */

169         return (ACPI_BTYPE_REGION | ACPI_BTYPE_BUFFER | ACPI_BTYPE_FIELD_UNIT);

171     case ARGI_DATAREFOBJ:

173         return (ACPI_BTYPE_INTEGER |ACPI_BTYPE_STRING | ACPI_BTYPE_BUFFER |
174             ACPI_BTYPE_PACKAGE | ACPI_BTYPE_REFERENCE | ACPI_BTYPE_DDB_HANDLE);

176     default:

178         break;
179     }

181     return (ACPI_BTYPE_OBJECTS_AND_REFS);
182 }


185 /*******************************************************************************
186  *
187  * FUNCTION:    AnMapEtypeToBtype
188  *
189  * PARAMETERS:  Etype               - Encoded ACPI Type
190  *
191  * RETURN:      Btype corresponding to the Etype
```



```
192  *
193  * DESCRIPTION: Convert an encoded ACPI type to a bitfield type applying the
194  *              operand conversion rules. In other words, returns the type(s)
195  *              this Etype is implicitly converted to during interpretation.
196  *
197  ******************************************************************************/

199 static UINT32
200 AnMapEtypeToBtype (
201     UINT32                  Etype)
202 {


205     if (Etype == ACPI_TYPE_ANY)
206     {
207         return (ACPI_BTYPE_OBJECTS_AND_REFS);
208     }

210     /* Try the standard ACPI data types */

212     if (Etype <= ACPI_TYPE_EXTERNAL_MAX)
213     {
214         /*
215          * This switch statement implements the allowed operand conversion
216          * rules as per the "ASL Data Types" section of the ACPI
217          * specification.
218          */
219         switch (Etype)
220         {
221         case ACPI_TYPE_INTEGER:

223             return (ACPI_BTYPE_COMPUTE_DATA | ACPI_BTYPE_DDB_HANDLE);

225         case ACPI_TYPE_STRING:
226         case ACPI_TYPE_BUFFER:

228             return (ACPI_BTYPE_COMPUTE_DATA);

230         case ACPI_TYPE_PACKAGE:

232             return (ACPI_BTYPE_PACKAGE);

234         case ACPI_TYPE_FIELD_UNIT:

236             return (ACPI_BTYPE_COMPUTE_DATA | ACPI_BTYPE_FIELD_UNIT);

238         case ACPI_TYPE_BUFFER_FIELD:

240             return (ACPI_BTYPE_COMPUTE_DATA | ACPI_BTYPE_BUFFER_FIELD);

242         case ACPI_TYPE_DDB_HANDLE:

244             return (ACPI_BTYPE_INTEGER | ACPI_BTYPE_DDB_HANDLE);

246         case ACPI_TYPE_DEBUG_OBJECT:

248             /* Cannot be used as a source operand */

250             return (0);

252         default:

254             return (1 << (Etype - 1));
255         }
256     }
```

```
258     /* Try the internal data types */

260     switch (Etype)
261     {
262     case ACPI_TYPE_LOCAL_REGION_FIELD:
263     case ACPI_TYPE_LOCAL_BANK_FIELD:
264     case ACPI_TYPE_LOCAL_INDEX_FIELD:

266         /* Named fields can be either Integer/Buffer/String */

268         return (ACPI_BTYPE_COMPUTE_DATA | ACPI_BTYPE_FIELD_UNIT);

270     case ACPI_TYPE_LOCAL_ALIAS:

272         return (ACPI_BTYPE_INTEGER);


275     case ACPI_TYPE_LOCAL_RESOURCE:
276     case ACPI_TYPE_LOCAL_RESOURCE_FIELD:

278         return (ACPI_BTYPE_REFERENCE);

280     default:

282         printf ("Unhandled encoded type: %X\n", Etype);
283         return (0);
284     }
285 }


288 /*******************************************************************************
289  *
290  * FUNCTION:    AnFormatBtype
291  *
292  * PARAMETERS:  Btype               - Bitfield of ACPI types
293  *              Buffer              - Where to put the ascii string
294  *
295  * RETURN:      None.
296  *
297  * DESCRIPTION: Convert a Btype to a string of ACPI types
298  *
299  ******************************************************************************/

301 void
302 AnFormatBtype (
303     char                    *Buffer,
304     UINT32                  Btype)
305 {
306     UINT32                  Type;
307     BOOLEAN                 First = TRUE;


310     *Buffer = 0;

312     if (Btype == 0)
313     {
314         strcat (Buffer, "NoReturnValue");
315         return;
316     }

318     for (Type = 1; Type <= ACPI_TYPE_EXTERNAL_MAX; Type++)
319     {
320         if (Btype & 0x00000001)
321         {
322             if (!First)
323             {
```



```
324                 strcat (Buffer, "|");
325             }
326             First = FALSE;
327             strcat (Buffer, AcpiUtGetTypeName (Type));
328         }
329         Btype >>= 1;
330     }

332     if (Btype & 0x00000001)
333     {
334         if (!First)
335         {
336             strcat (Buffer, "|");
337         }
338         First = FALSE;
339         strcat (Buffer, "Reference");
340     }

342     Btype >>= 1;
343     if (Btype & 0x00000001)
344     {
345         if (!First)
346         {
347             strcat (Buffer, "|");
348         }
349         First = FALSE;
350         strcat (Buffer, "Resource");
351     }
352 }


355 /*******************************************************************************
356  *
357  * FUNCTION:    AnGetBtype
358  *
359  * PARAMETERS:  Op                  - Parse node whose type will be returned.
360  *
361  * RETURN:      The Btype associated with the Op.
362  *
363  * DESCRIPTION: Get the (bitfield) ACPI type associated with the parse node.
364  *              Handles the case where the node is a name or method call and
365  *              the actual type must be obtained from the namespace node.
366  *
367  ******************************************************************************/

369 UINT32
370 AnGetBtype (
371     ACPI_PARSE_OBJECT       *Op)
372 {
373     ACPI_NAMESPACE_NODE     *Node;
374     ACPI_PARSE_OBJECT       *ReferencedNode;
375     UINT32                  ThisNodeBtype = 0;


378     if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)     ||
379         (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING)  ||
380         (Op->Asl.ParseOpcode == PARSEOP_METHODCALL))
381     {
382         Node = Op->Asl.Node;
383         if (!Node)
384         {
385             DbgPrint (ASL_DEBUG_OUTPUT,
386                 "No attached Nsnode: [%s] at line %u name [%s], ignoring typeche
387                 Op->Asl.ParseOpName, Op->Asl.LineNumber,
388                 Op->Asl.ExternalName);
389             return (ACPI_UINT32_MAX);
```

```
390         }

392         ThisNodeBtype = AnMapEtypeToBtype (Node->Type);
393         if (!ThisNodeBtype)
394         {
395             AslError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL, Op,
396                 "could not map type");
397         }

399         /*
400          * Since it was a named reference, enable the
401          * reference bit also
402          */
403         ThisNodeBtype |= ACPI_BTYPE_REFERENCE;

405         if (Op->Asl.ParseOpcode == PARSEOP_METHODCALL)
406         {
407             ReferencedNode = Node->Op;
408             if (!ReferencedNode)
409             {
410                 /* Check for an internal method */

412                 if (AnIsInternalMethod (Op))
413                 {
414                     return (AnGetInternalMethodReturnType (Op));
415                 }

417                 AslError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL, Op,
418                     "null Op pointer");
419                 return (ACPI_UINT32_MAX);
420             }

422             if (ReferencedNode->Asl.CompileFlags & NODE_METHOD_TYPED)
423             {
424                 ThisNodeBtype = ReferencedNode->Asl.AcpiBtype;
425             }
426             else
427             {
428                 return (ACPI_UINT32_MAX -1);
429             }
430         }
431     }
432     else
433     {
434         ThisNodeBtype = Op->Asl.AcpiBtype;
435     }

437     return (ThisNodeBtype);
438 }


441 /*******************************************************************************
442  *
443  * FUNCTION:    AnMapObjTypeToBtype
444  *
445  * PARAMETERS:  Op                  - A parse node
446  *
447  * RETURN:      A Btype
448  *
449  * DESCRIPTION: Map object to the associated "Btype"
450  *
451  ******************************************************************************/

453 UINT32
454 AnMapObjTypeToBtype (
455     ACPI_PARSE_OBJECT       *Op)
```



```
456 {

458     switch (Op->Asl.ParseOpcode)
459     {
460     case PARSEOP_OBJECTTYPE_BFF:        /* "BuffFieldObj" */

462         return (ACPI_BTYPE_BUFFER_FIELD);

464     case PARSEOP_OBJECTTYPE_BUF:        /* "BuffObj" */

466         return (ACPI_BTYPE_BUFFER);

468     case PARSEOP_OBJECTTYPE_DDB:        /* "DDBHandleObj" */

470         return (ACPI_BTYPE_DDB_HANDLE);

472     case PARSEOP_OBJECTTYPE_DEV:        /* "DeviceObj" */

474         return (ACPI_BTYPE_DEVICE);

476     case PARSEOP_OBJECTTYPE_EVT:        /* "EventObj" */

478         return (ACPI_BTYPE_EVENT);

480     case PARSEOP_OBJECTTYPE_FLD:        /* "FieldUnitObj" */

482         return (ACPI_BTYPE_FIELD_UNIT);

484     case PARSEOP_OBJECTTYPE_INT:        /* "IntObj" */

486         return (ACPI_BTYPE_INTEGER);

488     case PARSEOP_OBJECTTYPE_MTH:        /* "MethodObj" */

490         return (ACPI_BTYPE_METHOD);

492     case PARSEOP_OBJECTTYPE_MTX:        /* "MutexObj" */

494         return (ACPI_BTYPE_MUTEX);

496     case PARSEOP_OBJECTTYPE_OPR:        /* "OpRegionObj" */

498         return (ACPI_BTYPE_REGION);

500     case PARSEOP_OBJECTTYPE_PKG:        /* "PkgObj" */

502         return (ACPI_BTYPE_PACKAGE);

504     case PARSEOP_OBJECTTYPE_POW:        /* "PowerResObj" */

506         return (ACPI_BTYPE_POWER);

508     case PARSEOP_OBJECTTYPE_STR:        /* "StrObj" */

510         return (ACPI_BTYPE_STRING);

512     case PARSEOP_OBJECTTYPE_THZ:        /* "ThermalZoneObj" */

514         return (ACPI_BTYPE_THERMAL);

516     case PARSEOP_OBJECTTYPE_UNK:        /* "UnknownObj" */

518         return (ACPI_BTYPE_OBJECTS_AND_REFS);

520     default:
```

```
522         return (0);
523     }
524 }


527 #ifdef ACPI_OBSOLETE_FUNCTIONS
528 /*******************************************************************************
529  *
530  * FUNCTION:    AnMapBtypeToEtype
531  *
532  * PARAMETERS:  Btype               - Bitfield of ACPI types
533  *
534  * RETURN:      The Etype corresponding the the Btype
535  *
536  * DESCRIPTION: Convert a bitfield type to an encoded type
537  *
538  ******************************************************************************/

540 UINT32
541 AnMapBtypeToEtype (
542     UINT32              Btype)
543 {
544     UINT32              i;
545     UINT32              Etype;


548     if (Btype == 0)
549     {
550         return (0);
551     }

553     Etype = 1;
554     for (i = 1; i < Btype; i *= 2)
555     {
556         Etype++;
557     }

559     return (Etype);
560 }
561 #endif
```

```
**********************************************************
   15247 Thu Dec 26 13:48:26 2013
new/usr/src/common/acpica/compiler/aslcodegen.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslcodegen - AML code generation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslcodegen")

/* Local prototypes */

static ACPI_STATUS
CgAmlWriteWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static void
```



```
CgLocalWriteAmlData (
    ACPI_PARSE_OBJECT       *Op,
    void                    *Buffer,
    UINT32                  Length);

static void
CgWriteAmlOpcode (
    ACPI_PARSE_OBJECT       *Op);

static void
CgWriteTableHeader (
    ACPI_PARSE_OBJECT       *Op);

static void
CgCloseTable (
    void);

static void
CgWriteNode (
    ACPI_PARSE_OBJECT       *Op);


/******************************************************************************
 *
 * FUNCTION:    CgGenerateAmlOutput
 *
 * PARAMETERS:  None.
 *
 * RETURN:      None
 *
 * DESCRIPTION: Generate AML code. Currently generates the listing file
 *              simultaneously.
 *
 *****************************************************************************/

void
CgGenerateAmlOutput (
    void)
{

    DbgPrint (ASL_DEBUG_OUTPUT, "\nWriting AML\n\n");

    /* Generate the AML output file */

    FlSeekFile (ASL_FILE_SOURCE_OUTPUT, 0);
    Gbl_SourceLine = 0;
    Gbl_NextError = Gbl_ErrorLog;

    TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
        CgAmlWriteWalk, NULL, NULL);
    CgCloseTable ();
}


/******************************************************************************
 *
 * FUNCTION:    CgAmlWriteWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Parse tree walk to generate the AML code.
 *
 *****************************************************************************/
```

```
127 static ACPI_STATUS
128 CgAmlWriteWalk (
129     ACPI_PARSE_OBJECT       *Op,
130     UINT32                  Level,
131     void                    *Context)
132 {
133
134     /*
135      * Print header at level 0. Alignment assumes 32-bit pointers
136      */
137     if (!Level)
138     {
139         DbgPrint (ASL_TREE_OUTPUT,
140             "Final parse tree used for AML output:\n");
141         DbgPrint (ASL_TREE_OUTPUT,
142             "%*s Value    P_Op A_Op OpLen PByts Len  SubLen PSubLen OpPtr    Chi
143             76, " ");
144     }
145
146     /* Debug output */
147
148     DbgPrint (ASL_TREE_OUTPUT,
149         "%5.5d [%2d]", Op->Asl.LogicalLineNumber, Level);
150     UtPrintFormattedName (Op->Asl.ParseOpcode, Level);
151
152     if (Op->Asl.ParseOpcode == PARSEOP_NAMESEG    ||
153         Op->Asl.ParseOpcode == PARSEOP_NAMESTRING ||
154         Op->Asl.ParseOpcode == PARSEOP_METHODCALL)
155     {
156         DbgPrint (ASL_TREE_OUTPUT,
157             "%10.32s      ", Op->Asl.ExternalName);
158     }
159     else
160     {
161         DbgPrint (ASL_TREE_OUTPUT, "                ");
162     }
163
164     DbgPrint (ASL_TREE_OUTPUT,
165     "%08X %04X %04X %01X     %04X  %04X %04X   %04X    %08X %08X %08X %08X %08X
166             /* 1  */ (UINT32) Op->Asl.Value.Integer,
167             /* 2  */ Op->Asl.ParseOpcode,
168             /* 3  */ Op->Asl.AmlOpcode,
169             /* 4  */ Op->Asl.AmlOpcodeLength,
170             /* 5  */ Op->Asl.AmlPkgLenBytes,
171             /* 6  */ Op->Asl.AmlLength,
172             /* 7  */ Op->Asl.AmlSubtreeLength,
173             /* 8  */ Op->Asl.Parent ? Op->Asl.Parent->Asl.AmlSubtreeLength : 0,
174             /* 9  */ Op,
175             /* 10 */ Op->Asl.Child,
176             /* 11 */ Op->Asl.Parent,
177             /* 12 */ Op->Asl.CompileFlags,
178             /* 13 */ Op->Asl.AcpiBtype,
179             /* 14 */ Op->Asl.FinalAmlLength,
180             /* 15 */ Op->Asl.Column,
181             /* 16 */ Op->Asl.LineNumber);
182
183     /* Generate the AML for this node */
184
185     CgWriteNode (Op);
186     return (AE_OK);
187 }
188
189
190 /*******************************************************************************
191  *
192  * FUNCTION:    CgLocalWriteAmlData
```



```
193  *
194  * PARAMETERS:  Op              - Current parse op
195  *              Buffer          - Buffer to write
196  *              Length          - Size of data in buffer
197  *
198  * RETURN:      None
199  *
200  * DESCRIPTION: Write a buffer of AML data to the AML output file.
201  *
202  ******************************************************************************/
203
204 static void
205 CgLocalWriteAmlData (
206     ACPI_PARSE_OBJECT       *Op,
207     void                    *Buffer,
208     UINT32                  Length)
209 {
210
211     /* Write the raw data to the AML file */
212
213     FlWriteFile (ASL_FILE_AML_OUTPUT, Buffer, Length);
214
215     /* Update the final AML length for this node (used for listings) */
216
217     if (Op)
218     {
219         Op->Asl.FinalAmlLength += Length;
220     }
221 }
222
223
224 /*******************************************************************************
225  *
226  * FUNCTION:    CgWriteAmlOpcode
227  *
228  * PARAMETERS:  Op            - Parse node with an AML opcode
229  *
230  * RETURN:      None.
231  *
232  * DESCRIPTION: Write the AML opcode corresponding to a parse node.
233  *
234  ******************************************************************************/
235
236 static void
237 CgWriteAmlOpcode (
238     ACPI_PARSE_OBJECT       *Op)
239 {
240     UINT8                   PkgLenFirstByte;
241     UINT32                  i;
242     union {
243         UINT16                  Opcode;
244         UINT8                   OpcodeBytes[2];
245     } Aml;
246     union {
247         UINT32                  Len;
248         UINT8                   LenBytes[4];
249     } PkgLen;
250
251
252     /* We expect some DEFAULT_ARGs, just ignore them */
253
254     if (Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
255     {
256         return;
257     }
```

```
259     switch (Op->Asl.AmlOpcode)
260     {
261     case AML_UNASSIGNED_OPCODE:

263         /* These opcodes should not get here */

265         printf ("Found a node with an unassigned AML opcode\n");
266         FlPrintFile (ASL_FILE_STDERR, "Found a node with an unassigned AML opcod
267         return;

269     case AML_INT_RESERVEDFIELD_OP:

271         /* Special opcodes for within a field definition */

273         Aml.Opcode = AML_FIELD_OFFSET_OP;
274         break;

276     case AML_INT_ACCESSFIELD_OP:

278         Aml.Opcode = AML_FIELD_ACCESS_OP;
279         break;

281     case AML_INT_CONNECTION_OP:

283         Aml.Opcode = AML_FIELD_CONNECTION_OP;
284         break;

286     default:

288         Aml.Opcode = Op->Asl.AmlOpcode;
289         break;
290     }


293     switch (Aml.Opcode)
294     {
295     case AML_PACKAGE_LENGTH:

297         /* Value is the length to be encoded (Used in field definitions) */

299         PkgLen.Len = (UINT32) Op->Asl.Value.Integer;
300         break;

302     default:

304         /* Check for two-byte opcode */

306         if (Aml.Opcode > 0x00FF)
307         {
308             /* Write the high byte first */

310             CgLocalWriteAmlData (Op, &Aml.OpcodeBytes[1], 1);
311         }

313         CgLocalWriteAmlData (Op, &Aml.OpcodeBytes[0], 1);

315         /* Subtreelength doesn't include length of package length bytes */

317         PkgLen.Len = Op->Asl.AmlSubtreeLength + Op->Asl.AmlPkgLenBytes;
318         break;
319     }

321     /* Does this opcode have an associated "PackageLength" field? */

323     if (Op->Asl.CompileFlags & NODE_AML_PACKAGE)
324     {
```



```
325         if (Op->Asl.AmlPkgLenBytes == 1)
326         {
327             /* Simplest case -- no bytes to follow, just write the count */

329             CgLocalWriteAmlData (Op, &PkgLen.LenBytes[0], 1);
330         }
331         else if (Op->Asl.AmlPkgLenBytes != 0)
332         {
333             /*
334              * Encode the "bytes to follow" in the first byte, top two bits.
335              * The low-order nybble of the length is in the bottom 4 bits
336              */
337             PkgLenFirstByte = (UINT8)
338                 (((UINT32) (Op->Asl.AmlPkgLenBytes - 1) << 6) |
339                 (PkgLen.LenBytes[0] & 0x0F));

341             CgLocalWriteAmlData (Op, &PkgLenFirstByte, 1);

343             /*
344              * Shift the length over by the 4 bits we just stuffed
345              * in the first byte
346              */
347             PkgLen.Len >>= 4;

349             /* Now we can write the remaining bytes - either 1, 2, or 3 bytes */

351             for (i = 0; i < (UINT32) (Op->Asl.AmlPkgLenBytes - 1); i++)
352             {
353                 CgLocalWriteAmlData (Op, &PkgLen.LenBytes[i], 1);
354             }
355         }
356     }

358     switch (Aml.Opcode)
359     {
360     case AML_BYTE_OP:

362         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 1);
363         break;

365     case AML_WORD_OP:

367         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 2);
368        break;

370     case AML_DWORD_OP:

372         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 4);
373         break;

375     case AML_QWORD_OP:

377         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 8);
378         break;

380     case AML_STRING_OP:

382         CgLocalWriteAmlData (Op, Op->Asl.Value.String, Op->Asl.AmlLength);
383         break;

385     default:

387         /* All data opcodes must appear above */

389         break;
390     }
```

```
}


/*******************************************************************************
 *
 * FUNCTION:    CgWriteTableHeader
 *
 * PARAMETERS:  Op        - The DEFINITIONBLOCK node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write a table header corresponding to the DEFINITIONBLOCK
 *
 ******************************************************************************/

static void
CgWriteTableHeader (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Child;


    /* AML filename */

    Child = Op->Asl.Child;

    /* Signature */

    Child = Child->Asl.Next;
    strncpy (TableHeader.Signature, Child->Asl.Value.String, 4);

    /* Revision */

    Child = Child->Asl.Next;
    TableHeader.Revision = (UINT8) Child->Asl.Value.Integer;

    /* Command-line Revision override */

    if (Gbl_RevisionOverride)
    {
        TableHeader.Revision = Gbl_RevisionOverride;
    }

    /* OEMID */

    Child = Child->Asl.Next;
    strncpy (TableHeader.OemId, Child->Asl.Value.String, 6);

    /* OEM TableID */

    Child = Child->Asl.Next;
    strncpy (TableHeader.OemTableId, Child->Asl.Value.String, 8);

    /* OEM Revision */

    Child = Child->Asl.Next;
    TableHeader.OemRevision = (UINT32) Child->Asl.Value.Integer;

    /* Compiler ID */

    ACPI_MOVE_NAME (TableHeader.AslCompilerId, ASL_CREATOR_ID);

    /* Compiler version */

    TableHeader.AslCompilerRevision = ASL_REVISION;
```



```
    /* Table length. Checksum zero for now, will rewrite later */

    TableHeader.Length   = Gbl_TableLength;
    TableHeader.Checksum = 0;

    CgLocalWriteAmlData (Op, &TableHeader, sizeof (ACPI_TABLE_HEADER));
}


/*******************************************************************************
 *
 * FUNCTION:    CgCloseTable
 *
 * PARAMETERS:  None.
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Complete the ACPI table by calculating the checksum and
 *              re-writing the header.
 *
 ******************************************************************************/

static void
CgCloseTable (
    void)
{
    signed char         Sum;
    UINT8               FileByte;


    FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
    Sum = 0;

    /* Calculate the checksum over the entire file */

    while (FlReadFile (ASL_FILE_AML_OUTPUT, &FileByte, 1) == AE_OK)
    {
        Sum = (signed char) (Sum + FileByte);
    }

    /* Re-write the table header with the checksum */

    TableHeader.Checksum = (UINT8) (0 - Sum);

    FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
    CgLocalWriteAmlData (NULL, &TableHeader, sizeof (ACPI_TABLE_HEADER));
}


/*******************************************************************************
 *
 * FUNCTION:    CgWriteNode
 *
 * PARAMETERS:  Op            - Parse node to write.
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Write the AML that corresponds to a parse node.
 *
 ******************************************************************************/

static void
CgWriteNode (
    ACPI_PARSE_OBJECT       *Op)
{
    ASL_RESOURCE_NODE       *Rnode;
```

```
525     /* Always check for DEFAULT_ARG and other "Noop" nodes */
526     /* TBD: this may not be the best place for this check */
527
528     if ((Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)  ||
529         (Op->Asl.ParseOpcode == PARSEOP_EXTERNAL)     ||
530         (Op->Asl.ParseOpcode == PARSEOP_INCLUDE)      ||
531         (Op->Asl.ParseOpcode == PARSEOP_INCLUDE_END))
532     {
533         return;
534     }
535
536     Op->Asl.FinalAmlLength = 0;
537
538     switch (Op->Asl.AmlOpcode)
539     {
540     case AML_RAW_DATA_BYTE:
541     case AML_RAW_DATA_WORD:
542     case AML_RAW_DATA_DWORD:
543     case AML_RAW_DATA_QWORD:
544
545         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, Op->Asl.AmlLength);
546         return;
547
548
549     case AML_RAW_DATA_BUFFER:
550
551         CgLocalWriteAmlData (Op, Op->Asl.Value.Buffer, Op->Asl.AmlLength);
552         return;
553
554
555     case AML_RAW_DATA_CHAIN:
556
557         Rnode = ACPI_CAST_PTR (ASL_RESOURCE_NODE, Op->Asl.Value.Buffer);
558         while (Rnode)
559         {
560             CgLocalWriteAmlData (Op, Rnode->Buffer, Rnode->BufferLength);
561             Rnode = Rnode->Next;
562         }
563         return;
564
565     default:
566
567         /* Internal data opcodes must all appear above */
568
569         break;
570     }
571
572     switch (Op->Asl.ParseOpcode)
573     {
574     case PARSEOP_DEFAULT_ARG:
575
576         break;
577
578     case PARSEOP_DEFINITIONBLOCK:
579
580         CgWriteTableHeader (Op);
581         break;
582
583     case PARSEOP_NAMESEG:
584     case PARSEOP_NAMESTRING:
585     case PARSEOP_METHODCALL:
586
587         CgLocalWriteAmlData (Op, Op->Asl.Value.String, Op->Asl.AmlLength);
588         break;
```



```
589
590     default:
591
592         CgWriteAmlOpcode (Op);
593         break;
594     }
595 }
```

```
**********************************************************
   26878 Thu Dec 26 13:48:26 2013
new/usr/src/common/acpica/compiler/aslcompile.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslcompile - top level compile module
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"

#include <stdio.h>
#include <time.h>
#include <acapps.h>

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslcompile")

/*
 * Main parser entry
 * External is here in case the parser emits the same external in the
 * generated header. (Newer versions of Bison)
 */
int
AslCompilerparse(
```



```
    void);

/* Local prototypes */

static void
CmFlushSourceCode (
    void);

static void
FlConsumeAnsiComment (
    FILE                    *Handle,
    ASL_FILE_STATUS         *Status);

static void
FlConsumeNewComment (
    FILE                    *Handle,
    ASL_FILE_STATUS         *Status);

static void
CmDumpAllEvents (
    void);


/*******************************************************************************
 *
 * FUNCTION:    AslCompilerSignon
 *
 * PARAMETERS:  FileId      - ID of the output file
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display compiler signon
 *
 ******************************************************************************/

void
AslCompilerSignon (
    UINT32                  FileId)
{
    char                    *Prefix = "";
    char                    *UtilityName;


    /* Set line prefix depending on the destination file type */

    switch (FileId)
    {
    case ASL_FILE_ASM_SOURCE_OUTPUT:
    case ASL_FILE_ASM_INCLUDE_OUTPUT:

        Prefix = "; ";
        break;

    case ASL_FILE_HEX_OUTPUT:

        if (Gbl_HexOutputFlag == HEX_OUTPUT_ASM)
        {
            Prefix = "; ";
        }
        else if ((Gbl_HexOutputFlag == HEX_OUTPUT_C) ||
                 (Gbl_HexOutputFlag == HEX_OUTPUT_ASL))
        {
            FlPrintFile (ASL_FILE_HEX_OUTPUT, "/*\n");
            Prefix = " * ";
        }
        break;
```

```
128     case ASL_FILE_C_SOURCE_OUTPUT:
129     case ASL_FILE_C_OFFSET_OUTPUT:
130     case ASL_FILE_C_INCLUDE_OUTPUT:

132         Prefix = " * ";
133         break;

135     default:

137         /* No other output types supported */

139         break;
140     }

142     /* Running compiler or disassembler? */

144     if (Gbl_DisasmFlag)
145     {
146         UtilityName = AML_DISASSEMBLER_NAME;
147     }
148     else
149     {
150         UtilityName = ASL_COMPILER_NAME;
151     }

153     /* Compiler signon with copyright */

155     FlPrintFile (FileId, "%s\n", Prefix);
156     FlPrintFile (FileId, ACPI_COMMON_HEADER (UtilityName, Prefix));
157 }


160 /******************************************************************************
161  *
162  * FUNCTION:    AslCompilerFileHeader
163  *
164  * PARAMETERS:  FileId      - ID of the output file
165  *
166  * RETURN:      None
167  *
168  * DESCRIPTION: Header used at the beginning of output files
169  *
170  *****************************************************************************/

172 void
173 AslCompilerFileHeader (
174     UINT32                  FileId)
175 {
176     struct tm               *NewTime;
177     time_t                  Aclock;
178     char                    *Prefix = "";


181     /* Set line prefix depending on the destination file type */

183     switch (FileId)
184     {
185     case ASL_FILE_ASM_SOURCE_OUTPUT:
186     case ASL_FILE_ASM_INCLUDE_OUTPUT:

188         Prefix = "; ";
189         break;

191     case ASL_FILE_HEX_OUTPUT:
```



```
193         if (Gbl_HexOutputFlag == HEX_OUTPUT_ASM)
194         {
195             Prefix = "; ";
196         }
197         else if ((Gbl_HexOutputFlag == HEX_OUTPUT_C) ||
198                  (Gbl_HexOutputFlag == HEX_OUTPUT_ASL))
199         {
200             Prefix = " * ";
201         }
202         break;

204     case ASL_FILE_C_SOURCE_OUTPUT:
205     case ASL_FILE_C_OFFSET_OUTPUT:
206     case ASL_FILE_C_INCLUDE_OUTPUT:

208         Prefix = " * ";
209         break;

211     default:

213         /* No other output types supported */

215         break;
216     }

218     /* Compilation header with timestamp */

220     (void) time (&Aclock);
221     NewTime = localtime (&Aclock);

223     FlPrintFile (FileId,
224         "%sCompilation of \"%s\" - %s%s\n",
225         Prefix, Gbl_Files[ASL_FILE_INPUT].Filename, asctime (NewTime),
226         Prefix);

228     switch (FileId)
229     {
230     case ASL_FILE_C_SOURCE_OUTPUT:
231     case ASL_FILE_C_OFFSET_OUTPUT:
232     case ASL_FILE_C_INCLUDE_OUTPUT:

234         FlPrintFile (FileId, " */\n");
235         break;

237     default:

239         /* Nothing to do for other output types */

241         break;
242     }
243 }


246 /******************************************************************************
247  *
248  * FUNCTION:    CmFlushSourceCode
249  *
250  * PARAMETERS:  None
251  *
252  * RETURN:      None
253  *
254  * DESCRIPTION: Read in any remaining source code after the parse tree
255  *              has been constructed.
256  *
257  *****************************************************************************/
```

```
static void
CmFlushSourceCode (
    void)
{
    char                    Buffer;


    while (FlReadFile (ASL_FILE_INPUT, &Buffer, 1) != AE_ERROR)
    {
        AslInsertLineBuffer ((int) Buffer);
    }

    AslResetCurrentLineBuffer ();
}


/*******************************************************************************
 *
 * FUNCTION:    FlConsume*
 *
 * PARAMETERS:  Handle              - Open input file
 *              Status              - File current status struct
 *
 * RETURN:      Number of lines consumed
 *
 * DESCRIPTION: Step over both types of comment during check for ascii chars
 *
 ******************************************************************************/

static void
FlConsumeAnsiComment (
    FILE                    *Handle,
    ASL_FILE_STATUS         *Status)
{
    UINT8                   Byte;
    BOOLEAN                 ClosingComment = FALSE;


    while (fread (&Byte, 1, 1, Handle) == 1)
    {
        /* Scan until comment close is found */

        if (ClosingComment)
        {
            if (Byte == '/')
            {
                return;
            }

            if (Byte != '*')
            {
                /* Reset */

                ClosingComment = FALSE;
            }
        }
        else if (Byte == '*')
        {
            ClosingComment = TRUE;
        }

        /* Maintain line count */

        if (Byte == 0x0A)
        {
            Status->Line++;
```



```
        }

        Status->Offset++;
    }
}


static void
FlConsumeNewComment (
    FILE                    *Handle,
    ASL_FILE_STATUS         *Status)
{
    UINT8                   Byte;


    while (fread (&Byte, 1, 1, Handle) == 1)
    {
        Status->Offset++;

        /* Comment ends at newline */

        if (Byte == 0x0A)
        {
            Status->Line++;
            return;
        }
    }
}


/*******************************************************************************
 *
 * FUNCTION:    FlCheckForAcpiTable
 *
 * PARAMETERS:  Handle              - Open input file
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Determine if a file seems to be a binary ACPI table, via the
 *              following checks on what would be the table header:
 *              0) File must be at least as long as an ACPI_TABLE_HEADER
 *              1) The header length field must match the file size
 *              2) Signature, OemId, OemTableId, AslCompilerId must be ASCII
 *
 ******************************************************************************/

ACPI_STATUS
FlCheckForAcpiTable (
    FILE                    *Handle)
{
    ACPI_TABLE_HEADER       Table;
    UINT32                  FileSize;
    size_t                  Actual;
    UINT32                  i;


    /* Read a potential table header */

    Actual = fread (&Table, 1, sizeof (ACPI_TABLE_HEADER), Handle);
    fseek (Handle, 0, SEEK_SET);

    if (Actual < sizeof (ACPI_TABLE_HEADER))
    {
        return (AE_ERROR);
    }
```

```
391     /* Header length field must match the file size */
393     FileSize = DtGetFileSize (Handle);
394     if (Table.Length != FileSize)
395     {
396         return (AE_ERROR);
397     }
399     /*
400      * These fields must be ASCII:
401      * Signature, OemId, OemTableId, AslCompilerId.
402      * We allow a NULL terminator in OemId and OemTableId.
403      */
404     for (i = 0; i < ACPI_NAME_SIZE; i++)
405     {
406         if (!ACPI_IS_ASCII ((UINT8) Table.Signature[i]))
407         {
408             return (AE_ERROR);
409         }
411         if (!ACPI_IS_ASCII ((UINT8) Table.AslCompilerId[i]))
412         {
413             return (AE_ERROR);
414         }
415     }
417     for (i = 0; (i < ACPI_OEM_ID_SIZE) && (Table.OemId[i]); i++)
418     {
419         if (!ACPI_IS_ASCII ((UINT8) Table.OemId[i]))
420         {
421             return (AE_ERROR);
422         }
423     }
425     for (i = 0; (i < ACPI_OEM_TABLE_ID_SIZE) && (Table.OemTableId[i]); i++)
426     {
427         if (!ACPI_IS_ASCII ((UINT8) Table.OemTableId[i]))
428         {
429             return (AE_ERROR);
430         }
431     }
433     printf ("Binary file appears to be a valid ACPI table, disassembling\n");
434     return (AE_OK);
435 }
438 /******************************************************************************
439  *
440  * FUNCTION:    FlCheckForAscii
441  *
442  * PARAMETERS:  Handle              - Open input file
443  *              Filename            - Input filename
444  *              DisplayErrors       - TRUE if error messages desired
445  *
446  * RETURN:      Status
447  *
448  * DESCRIPTION: Verify that the input file is entirely ASCII. Ignores characters
449  *              within comments. Note: does not handle nested comments and does
450  *              not handle comment delimiters within string literals. However,
451  *              on the rare chance this happens and an invalid character is
452  *              missed, the parser will catch the error by failing in some
453  *              spectactular manner.
454  *
455  *****************************************************************************/
```



```
457 ACPI_STATUS
458 FlCheckForAscii (
459     FILE                    *Handle,
460     char                    *Filename,
461     BOOLEAN                 DisplayErrors)
462 {
463     UINT8                   Byte;
464     ACPI_SIZE               BadBytes = 0;
465     BOOLEAN                 OpeningComment = FALSE;
466     ASL_FILE_STATUS         Status;
469     Status.Line = 1;
470     Status.Offset = 0;
472     /* Read the entire file */
474     while (fread (&Byte, 1, 1, Handle) == 1)
475     {
476         /* Ignore comment fields (allow non-ascii within) */
478         if (OpeningComment)
479         {
480             /* Check for second comment open delimiter */
482             if (Byte == '*')
483             {
484                 FlConsumeAnsiComment (Handle, &Status);
485             }
487             if (Byte == '/')
488             {
489                 FlConsumeNewComment (Handle, &Status);
490             }
492             /* Reset */
494             OpeningComment = FALSE;
495         }
496         else if (Byte == '/')
497         {
498             OpeningComment = TRUE;
499         }
501         /* Check for an ASCII character */
503         if (!ACPI_IS_ASCII (Byte))
504         {
505             if ((BadBytes < 10) && (DisplayErrors))
506             {
507                 AcpiOsPrintf (
508                     "Non-ASCII character [0x%2.2X] found in line %u, file offset
509                     Byte, Status.Line, Status.Offset);
510             }
512             BadBytes++;
513         }
515         /* Update line counter */
517         else if (Byte == 0x0A)
518         {
519             Status.Line++;
520         }
522         Status.Offset++;
```

```
523     }
525     /* Seek back to the beginning of the source file */
527     fseek (Handle, 0, SEEK_SET);
529     /* Were there any non-ASCII characters in the file? */
531     if (BadBytes)
532     {
533         if (DisplayErrors)
534         {
535             AcpiOsPrintf (
536                 "%u non-ASCII characters found in input source text, could be a
537                 BadBytes);
538             AslError (ASL_ERROR, ASL_MSG_NON_ASCII, NULL, Filename);
539         }
541         return (AE_BAD_CHARACTER);
542     }
544     /* File is OK (100% ASCII) */
546     return (AE_OK);
547 }


550 /******************************************************************************
551  *
552  * FUNCTION:    CmDoCompile
553  *
554  * PARAMETERS:  None
555  *
556  * RETURN:      Status (0 = OK)
557  *
558  * DESCRIPTION: This procedure performs the entire compile
559  *
560  *****************************************************************************/

562 int
563 CmDoCompile (
564     void)
565 {
566     ACPI_STATUS             Status;
567     UINT8                   FullCompile;
568     UINT8                   Event;


571     FullCompile = UtBeginEvent ("*** Total Compile time ***");
572     Event = UtBeginEvent ("Open input and output files");
573     UtEndEvent (Event);

575     Event = UtBeginEvent ("Preprocess input file");
576     if (Gbl_PreprocessFlag)
577     {
578         /* Preprocessor */

580         PrDoPreprocess ();
581         if (Gbl_PreprocessOnly)
582         {
583             UtEndEvent (Event);
584             CmCleanupAndExit ();
585             return (0);
586         }
587     }
588     UtEndEvent (Event);
```



```
590     /* Build the parse tree */

592     Event = UtBeginEvent ("Parse source code and build parse tree");
593     AslCompilerparse();
594     UtEndEvent (Event);

596     /* Check for parse errors */

598     Status = AslCheckForErrorExit ();
599     if (ACPI_FAILURE (Status))
600     {
601         fprintf (stderr, "Compiler aborting due to parser-detected syntax error(
602         LsDumpParseTree ();
603         goto ErrorExit;
604     }

606     /* Did the parse tree get successfully constructed? */

608     if (!RootNode)
609     {
610         /*
611          * If there are no errors, then we have some sort of
612          * internal problem.
613          */
614         AslError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL,
615             NULL, "- Could not resolve parse tree root node");

617         goto ErrorExit;
618     }


621     /* Flush out any remaining source after parse tree is complete */

623     Event = UtBeginEvent ("Flush source input");
624     CmFlushSourceCode ();

626     /* Optional parse tree dump, compiler debug output only */

628     LsDumpParseTree ();

630     OpcGetIntegerWidth (RootNode);
631     UtEndEvent (Event);

633     /* Pre-process parse tree for any operator transforms */

635     Event = UtBeginEvent ("Parse tree transforms");
636     DbgPrint (ASL_DEBUG_OUTPUT, "\nParse tree transforms\n\n");
637     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
638         TrAmlTransformWalk, NULL, NULL);
639     UtEndEvent (Event);

641     /* Generate AML opcodes corresponding to the parse tokens */

643     Event = UtBeginEvent ("Generate AML opcodes");
644     DbgPrint (ASL_DEBUG_OUTPUT, "\nGenerating AML opcodes\n\n");
645     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD, NULL,
646         OpcAmlOpcodeWalk, NULL);
647     UtEndEvent (Event);

649     /*
650      * Now that the input is parsed, we can open the AML output file.
651      * Note: by default, the name of this file comes from the table descriptor
652      * within the input file.
653      */
654     Event = UtBeginEvent ("Open AML output file");
```

```
655     Status = FlOpenAmlOutputFile (Gbl_OutputFilenamePrefix);
656     UtEndEvent (Event);
657     if (ACPI_FAILURE (Status))
658     {
659         AePrintErrorLog (ASL_FILE_STDERR);
660         return (-1);
661     }

663     /* Interpret and generate all compile-time constants */

665     Event = UtBeginEvent ("Constant folding via AML interpreter");
666     DbgPrint (ASL_DEBUG_OUTPUT,
667         "\nInterpreting compile-time constant expressions\n\n");
668     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
669         OpcAmlConstantWalk, NULL, NULL);
670     UtEndEvent (Event);

672     /* Update AML opcodes if necessary, after constant folding */

674     Event = UtBeginEvent ("Updating AML opcodes after constant folding");
675     DbgPrint (ASL_DEBUG_OUTPUT,
676         "\nUpdating AML opcodes after constant folding\n\n");
677     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD,
678         NULL, OpcAmlOpcodeUpdateWalk, NULL);
679     UtEndEvent (Event);

681     /* Calculate all AML package lengths */

683     Event = UtBeginEvent ("Generate AML package lengths");
684     DbgPrint (ASL_DEBUG_OUTPUT, "\nGenerating Package lengths\n\n");
685     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD, NULL,
686         LnPackageLengthWalk, NULL);
687     UtEndEvent (Event);

689     if (Gbl_ParseOnlyFlag)
690     {
691         AePrintErrorLog (ASL_FILE_STDERR);
692         UtDisplaySummary (ASL_FILE_STDERR);
693         if (Gbl_DebugFlag)
694         {
695             /* Print error summary to the stdout also */

697             AePrintErrorLog (ASL_FILE_STDOUT);
698             UtDisplaySummary (ASL_FILE_STDOUT);
699         }
700         UtEndEvent (FullCompile);
701         return (0);
702     }

704     /*
705      * Create an internal namespace and use it as a symbol table
706      */

708     /* Namespace loading */

710     Event = UtBeginEvent ("Create ACPI Namespace");
711     Status = LdLoadNamespace (RootNode);
712     UtEndEvent (Event);
713     if (ACPI_FAILURE (Status))
714     {
715         goto ErrorExit;
716     }

718     /* Namespace cross-reference */

720     AslGbl_NamespaceEvent = UtBeginEvent ("Cross reference parse tree and Namesp
```



```
721     Status = XfCrossReferenceNamespace ();
722     if (ACPI_FAILURE (Status))
723     {
724         goto ErrorExit;
725     }

727     /* Namespace - Check for non-referenced objects */

729     LkFindUnreferencedObjects ();
730     UtEndEvent (AslGbl_NamespaceEvent);

732     /*
733      * Semantic analysis. This can happen only after the
734      * namespace has been loaded and cross-referenced.
735      *
736      * part one - check control methods
737      */
738     Event = UtBeginEvent ("Analyze control method return types");
739     AnalysisWalkInfo.MethodStack = NULL;

741     DbgPrint (ASL_DEBUG_OUTPUT, "\nSemantic analysis - Method analysis\n\n");
742     TrWalkParseTree (RootNode, ASL_WALK_VISIT_TWICE,
743         MtMethodAnalysisWalkBegin,
744         MtMethodAnalysisWalkEnd, &AnalysisWalkInfo);
745     UtEndEvent (Event);

747     /* Semantic error checking part two - typing of method returns */

749     Event = UtBeginEvent ("Determine object types returned by methods");
750     DbgPrint (ASL_DEBUG_OUTPUT, "\nSemantic analysis - Method typing\n\n");
751     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD,
752         NULL, AnMethodTypingWalkEnd, NULL);
753     UtEndEvent (Event);

755     /* Semantic error checking part three - operand type checking */

757     Event = UtBeginEvent ("Analyze AML operand types");
758     DbgPrint (ASL_DEBUG_OUTPUT, "\nSemantic analysis - Operand type checking\n\n
759     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD,
760         NULL, AnOperandTypecheckWalkEnd, &AnalysisWalkInfo);
761     UtEndEvent (Event);

763     /* Semantic error checking part four - other miscellaneous checks */

765     Event = UtBeginEvent ("Miscellaneous analysis");
766     DbgPrint (ASL_DEBUG_OUTPUT, "\nSemantic analysis - miscellaneous\n\n");
767     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
768         AnOtherSemanticAnalysisWalkBegin,
769         NULL, &AnalysisWalkInfo);
770     UtEndEvent (Event);

772     /* Calculate all AML package lengths */

774     Event = UtBeginEvent ("Finish AML package length generation");
775     DbgPrint (ASL_DEBUG_OUTPUT, "\nGenerating Package lengths\n\n");
776     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD, NULL,
777         LnInitLengthsWalk, NULL);
778     TrWalkParseTree (RootNode, ASL_WALK_VISIT_UPWARD, NULL,
779         LnPackageLengthWalk, NULL);
780     UtEndEvent (Event);

782     /* Code generation - emit the AML */

784     Event = UtBeginEvent ("Generate AML code and write output files");
785     CgGenerateAmlOutput ();
786     UtEndEvent (Event);
```

```
788     Event = UtBeginEvent ("Write optional output files");
789     CmDoOutputFiles ();
790     UtEndEvent (Event);

792     UtEndEvent (FullCompile);
793     CmCleanupAndExit ();
794     return (0);

796 ErrorExit:
797     UtEndEvent (FullCompile);
798     CmCleanupAndExit ();
799     return (-1);
800 }


803 /******************************************************************************
804  *
805  * FUNCTION:    CmDoOutputFiles
806  *
807  * PARAMETERS:  None
808  *
809  * RETURN:      None.
810  *
811  * DESCRIPTION: Create all "listing" type files
812  *
813  *****************************************************************************/

815 void
816 CmDoOutputFiles (
817     void)
818 {

820     /* Create listings and hex files */

822     LsDoListings ();
823     HxDoHexOutput ();

825     /* Dump the namespace to the .nsp file if requested */

827     (void) NsDisplayNamespace ();
828 }


831 /******************************************************************************
832  *
833  * FUNCTION:    CmDumpAllEvents
834  *
835  * PARAMETERS:  None
836  *
837  * RETURN:      None.
838  *
839  * DESCRIPTION: Dump all compiler events
840  *
841  *****************************************************************************/

843 static void
844 CmDumpAllEvents (
845     void)
846 {
847     ASL_EVENT_INFO          *Event;
848     UINT32                  Delta;
849     UINT32                  USec;
850     UINT32                  MSec;
851     UINT32                  i;
```



```
854     Event = AslGbl_Events;

856     DbgPrint (ASL_DEBUG_OUTPUT, "\n\nElapsed time for major events\n\n");
857     if (Gbl_CompileTimesFlag)
858     {
859         printf ("\nElapsed time for major events\n\n");
860     }

862     for (i = 0; i < AslGbl_NextEvent; i++)
863     {
864         if (Event->Valid)
865         {
866             /* Delta will be in 100-nanosecond units */

868             Delta = (UINT32) (Event->EndTime - Event->StartTime);

870             USec = Delta / ACPI_100NSEC_PER_USEC;
871             MSec = Delta / ACPI_100NSEC_PER_MSEC;

873             /* Round milliseconds up */

875             if ((USec - (MSec * ACPI_USEC_PER_MSEC)) >= 500)
876             {
877                 MSec++;
878             }

880             DbgPrint (ASL_DEBUG_OUTPUT, "%8u usec %8u msec - %s\n",
881                 USec, MSec, Event->EventName);

883             if (Gbl_CompileTimesFlag)
884             {
885                 printf ("%8u usec %8u msec - %s\n",
886                     USec, MSec, Event->EventName);
887             }
888         }

890         Event++;
891     }
892 }


895 /******************************************************************************
896  *
897  * FUNCTION:    CmCleanupAndExit
898  *
899  * PARAMETERS:  None
900  *
901  * RETURN:      None.
902  *
903  * DESCRIPTION: Close all open files and exit the compiler
904  *
905  *****************************************************************************/

907 void
908 CmCleanupAndExit (
909     void)
910 {
911     UINT32                  i;
912     BOOLEAN                 DeleteAmlFile = FALSE;


915     AePrintErrorLog (ASL_FILE_STDERR);
916     if (Gbl_DebugFlag)
917     {
918         /* Print error summary to stdout also */
```

```
920         AePrintErrorLog (ASL_FILE_STDOUT);
921     }

923     /* Emit compile times if enabled */

925     CmDumpAllEvents ();

927     if (Gbl_CompileTimesFlag)
928     {
929         printf ("\nMiscellaneous compile statistics\n\n");
930         printf ("%11u : %s\n", TotalParseNodes, "Parse nodes");
931         printf ("%11u : %s\n", Gbl_NsLookupCount, "Namespace searches");
932         printf ("%11u : %s\n", TotalNamedObjects, "Named objects");
933         printf ("%11u : %s\n", TotalMethods, "Control methods");
934         printf ("%11u : %s\n", TotalAllocations, "Memory Allocations");
935         printf ("%11u : %s\n", TotalAllocated, "Total allocated memory");
936         printf ("%11u : %s\n", TotalFolds, "Constant subtrees folded");
937         printf ("\n");
938     }

940     if (Gbl_NsLookupCount)
941     {
942         DbgPrint (ASL_DEBUG_OUTPUT,
943             "\n\nMiscellaneous compile statistics\n\n");

945         DbgPrint (ASL_DEBUG_OUTPUT,
946             "%32s : %u\n", "Total Namespace searches",
947             Gbl_NsLookupCount);

949         DbgPrint (ASL_DEBUG_OUTPUT,
950             "%32s : %u usec\n", "Time per search", ((UINT32)
951             (AslGbl_Events[AslGbl_NamespaceEvent].EndTime -
952                 AslGbl_Events[AslGbl_NamespaceEvent].StartTime) / 10) /
953                 Gbl_NsLookupCount);
954     }

956     if (Gbl_ExceptionCount[ASL_ERROR] > ASL_MAX_ERROR_COUNT)
957     {
958         printf ("\nMaximum error count (%u) exceeded\n",
959             ASL_MAX_ERROR_COUNT);
960     }

962     UtDisplaySummary (ASL_FILE_STDOUT);

964     /*
965      * We will delete the AML file if there are errors and the
966      * force AML output option has not been used.
967      */
968     if ((Gbl_ExceptionCount[ASL_ERROR] > 0) && (!Gbl_IgnoreErrors) &&
969         Gbl_Files[ASL_FILE_AML_OUTPUT].Handle)
970     {
971         DeleteAmlFile = TRUE;
972     }

974     /* Close all open files */

976     /*
977      * Take care with the preprocessor file (.i), it might be the same
978      * as the "input" file, depending on where the compiler has terminated
979      * or aborted. Prevent attempt to close the same file twice in
980      * loop below.
981      */
982     if (Gbl_Files[ASL_FILE_PREPROCESSOR].Handle ==
983         Gbl_Files[ASL_FILE_INPUT].Handle)
984     {
```



```
985         Gbl_Files[ASL_FILE_PREPROCESSOR].Handle = NULL;
986     }

988     /* Close the standard I/O files */

990     for (i = ASL_FILE_INPUT; i < ASL_MAX_FILE_TYPE; i++)
991     {
992         FlCloseFile (i);
993     }

995     /* Delete AML file if there are errors */

997     if (DeleteAmlFile)
998     {
999         FlDeleteFile (ASL_FILE_AML_OUTPUT);
1000    }

1002    /* Delete the preprocessor output file (.i) unless -li flag is set */

1004    if (!Gbl_PreprocessorOutputFlag &&
1005        Gbl_PreprocessFlag)
1006    {
1007        FlDeleteFile (ASL_FILE_PREPROCESSOR);
1008    }

1010    /*
1011     * Delete intermediate ("combined") source file (if -ls flag not set)
1012     * This file is created during normal ASL/AML compiles. It is not
1013     * created by the data table compiler.
1014     *
1015     * If the -ls flag is set, then the .SRC file should not be deleted.
1016     * In this case, Gbl_SourceOutputFlag is set to TRUE.
1017     *
1018     * Note: Handles are cleared by FlCloseFile above, so we look at the
1019     * filename instead, to determine if the .SRC file was actually
1020     * created.
1021     *
1022     * TBD: SourceOutput should be .TMP, then rename if we want to keep it?
1023     */
1024    if (!Gbl_SourceOutputFlag)
1025    {
1026        FlDeleteFile (ASL_FILE_SOURCE_OUTPUT);
1027    }
1028 }
```

```
**********************************************************
   25960 Thu Dec 26 13:48:26 2013
new/usr/src/common/acpica/compiler/aslcompiler.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslcompiler.h - common include file for iASL
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ASLCOMPILER_H
#define __ASLCOMPILER_H

#include "acpi.h"
#include "accommon.h"
#include "amlresrc.h"
#include "acdebug.h"

/* Microsoft-specific */

#if (defined WIN32 || defined WIN64)

/* warn : used #pragma pack */
#pragma warning(disable:4103)
```



```
/* warn : named type definition in parentheses */
#pragma warning(disable:4115)
#endif

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>

/* Compiler headers */

#include "asldefine.h"
#include "asltypes.h"
#include "aslmessages.h"
#include "aslglobal.h"
#include "preprocess.h"


/******************************************************************************
 *
 * Compiler prototypes
 *
 *****************************************************************************/

/*
 * Main ASL parser - generated from flex/bison, lex/yacc, etc.
 */
ACPI_PARSE_OBJECT *
AslDoError (
    void);

int
AslCompilerlex(
    void);

void
AslResetCurrentLineBuffer (
    void);

void
AslInsertLineBuffer (
    int                     SourceChar);

int
AslPopInputFileStack (
    void);

void
AslPushInputFileStack (
    FILE                    *InputFile,
    char                    *Filename);

/*
 * aslstartup - entered from main()
 */
void
AslInitializeGlobals (
    void);

typedef
ACPI_STATUS (*ASL_PATHNAME_CALLBACK) (
    char *);

ACPI_STATUS
```

```
126 AslDoOneFile (
127     char                    *Filename);

129 ACPI_STATUS
130 AslCheckForErrorExit (
131     void);


134 /*
135  * aslcompile - compile mainline
136  */
137 void
138 AslCompilerSignon (
139     UINT32                  FileId);

141 void
142 AslCompilerFileHeader (
143     UINT32                  FileId);

145 int
146 CmDoCompile (
147     void);

149 void
150 CmDoOutputFiles (
151     void);

153 void
154 CmCleanupAndExit (
155     void);

157 ACPI_STATUS
158 FlCheckForAcpiTable (
159     FILE                    *Handle);

161 ACPI_STATUS
162 FlCheckForAscii (
163     FILE                    *Handle,
164     char                    *Filename,
165     BOOLEAN                 DisplayErrors);


168 /*
169  * aslwalks - semantic analysis and parse tree walks
170  */
171 ACPI_STATUS
172 AnOtherSemanticAnalysisWalkBegin (
173     ACPI_PARSE_OBJECT       *Op,
174     UINT32                  Level,
175     void                    *Context);

177 ACPI_STATUS
178 AnOtherSemanticAnalysisWalkEnd (
179     ACPI_PARSE_OBJECT       *Op,
180     UINT32                  Level,
181     void                    *Context);

183 ACPI_STATUS
184 AnOperandTypecheckWalkEnd (
185     ACPI_PARSE_OBJECT       *Op,
186     UINT32                  Level,
187     void                    *Context);

189 ACPI_STATUS
190 AnMethodTypingWalkEnd (
191     ACPI_PARSE_OBJECT       *Op,
```



```
192     UINT32                  Level,
193     void                    *Context);


196 /*
197  * aslmethod - Control method analysis walk
198  */
199 ACPI_STATUS
200 MtMethodAnalysisWalkBegin (
201     ACPI_PARSE_OBJECT       *Op,
202     UINT32                  Level,
203     void                    *Context);

205 ACPI_STATUS
206 MtMethodAnalysisWalkEnd (
207     ACPI_PARSE_OBJECT       *Op,
208     UINT32                  Level,
209     void                    *Context);


212 /*
213  * aslbtypes - bitfield data types
214  */
215 UINT32
216 AnMapObjTypeToBtype (
217     ACPI_PARSE_OBJECT       *Op);

219 UINT32
220 AnMapArgTypeToBtype (
221     UINT32                  ArgType);

223 UINT32
224 AnGetBtype (
225     ACPI_PARSE_OBJECT       *Op);

227 void
228 AnFormatBtype (
229     char                    *Buffer,
230     UINT32                  Btype);


233 /*
234  * aslanalyze - Support functions for parse tree walks
235  */
236 void
237 AnCheckId (
238     ACPI_PARSE_OBJECT       *Op,
239     ACPI_NAME               Type);

241 /* Values for Type argument above */

243 #define ASL_TYPE_HID        0
244 #define ASL_TYPE_CID        1

246 BOOLEAN
247 AnIsInternalMethod (
248     ACPI_PARSE_OBJECT       *Op);

250 UINT32
251 AnGetInternalMethodReturnType (
252     ACPI_PARSE_OBJECT       *Op);

254 BOOLEAN
255 AnLastStatementIsReturn (
256     ACPI_PARSE_OBJECT       *Op);
```

```
258 void
259 AnCheckMethodReturnValue (
260     ACPI_PARSE_OBJECT       *Op,
261     const ACPI_OPCODE_INFO  *OpInfo,
262     ACPI_PARSE_OBJECT       *ArgOp,
263     UINT32                  RequiredBtypes,
264     UINT32                  ThisNodeBtype);

266 BOOLEAN
267 AnIsResultUsed (
268     ACPI_PARSE_OBJECT       *Op);

270 void
271 ApCheckForGpeNameConflict (
272     ACPI_PARSE_OBJECT       *Op);

274 void
275 ApCheckRegMethod (
276     ACPI_PARSE_OBJECT       *Op);


279 /*
280  * aslerror - error handling/reporting
281  */
282 void
283 AslError (
284     UINT8                   Level,
285     UINT8                   MessageId,
286     ACPI_PARSE_OBJECT       *Op,
287     char                    *ExtraMessage);

289 ACPI_STATUS
290 AslDisableException (
291     char                    *MessageIdString);

293 BOOLEAN
294 AslIsExceptionDisabled (
295     UINT8                   Level,
296     UINT8                   MessageId);

298 void
299 AslCoreSubsystemError (
300     ACPI_PARSE_OBJECT       *Op,
301     ACPI_STATUS             Status,
302     char                    *ExtraMessage,
303     BOOLEAN                 Abort);

305 int
306 AslCompilererror(
307     const char              *s);

309 void
310 AslCommonError (
311     UINT8                   Level,
312     UINT8                   MessageId,
313     UINT32                  CurrentLineNumber,
314     UINT32                  LogicalLineNumber,
315     UINT32                  LogicalByteOffset,
316     UINT32                  Column,
317     char                    *Filename,
318     char                    *ExtraMessage);

320 void
321 AslCommonError2 (
322     UINT8                   Level,
323     UINT8                   MessageId,
```



```
324     UINT32                  LineNumber,
325     UINT32                  Column,
326     char                    *SourceLine,
327     char                    *Filename,
328     char                    *ExtraMessage);

330 void
331 AePrintException (
332     UINT32                  FileId,
333     ASL_ERROR_MSG           *Enode,
334     char                    *Header);

336 void
337 AePrintErrorLog (
338     UINT32                  FileId);

340 void
341 AeClearErrorLog (
342     void);

344 ACPI_PHYSICAL_ADDRESS
345 AeLocalGetRootPointer (
346     void);


349 /*
350  * asllisting - generate all "listing" type files
351  */
352 void
353 LsDoListings (
354     void);

356 void
357 LsWriteNodeToAsmListing (
358     ACPI_PARSE_OBJECT       *Op);

360 void
361 LsWriteNode (
362     ACPI_PARSE_OBJECT       *Op,
363     UINT32                  FileId);

365 void
366 LsDumpParseTree (
367     void);


370 /*
371  * asllistsup - Listing file support utilities
372  */
373 void
374 LsDumpAscii (
375     UINT32                  FileId,
376     UINT32                  Count,
377     UINT8                   *Buffer);

379 void
380 LsDumpAsciiInComment (
381     UINT32                  FileId,
382     UINT32                  Count,
383     UINT8                   *Buffer);

385 void
386 LsCheckException (
387     UINT32                  LineNumber,
388     UINT32                  FileId);
```

```
390 void
391 LsFlushListingBuffer (
392     UINT32                  FileId);

394 void
395 LsWriteListingHexBytes (
396     UINT8                   *Buffer,
397     UINT32                  Length,
398     UINT32                  FileId);

400 void
401 LsWriteSourceLines (
402     UINT32                  ToLineNumber,
403     UINT32                  ToLogicalLineNumber,
404     UINT32                  FileId);

406 UINT32
407 LsWriteOneSourceLine (
408     UINT32                  FileId);

410 void
411 LsPushNode (
412     char                    *Filename);

414 ASL_LISTING_NODE *
415 LsPopNode (
416     void);


419 /*
420  * aslhex - generate all "hex" output files (C, ASM, ASL)
421  */
422 void
423 HxDoHexOutput (
424     void);


427 /*
428  * aslfold - constant folding
429  */
430 ACPI_STATUS
431 OpcAmlConstantWalk (
432     ACPI_PARSE_OBJECT       *Op,
433     UINT32                  Level,
434     void                    *Context);


437 /*
438  * asloffset - generate C offset file for BIOS support
439  */
440 ACPI_STATUS
441 LsAmlOffsetWalk (
442     ACPI_PARSE_OBJECT       *Op,
443     UINT32                  Level,
444     void                    *Context);

446 void
447 LsDoOffsetTableHeader (
448     UINT32                  FileId);

450 void
451 LsDoOffsetTableFooter (
452     UINT32                  FileId);


455 /*
```



```
456  * aslopcodes - generate AML opcodes
457  */
458 ACPI_STATUS
459 OpcAmlOpcodeWalk (
460     ACPI_PARSE_OBJECT       *Op,
461     UINT32                  Level,
462     void                    *Context);

464 ACPI_STATUS
465 OpcAmlOpcodeUpdateWalk (
466     ACPI_PARSE_OBJECT       *Op,
467     UINT32                  Level,
468     void                    *Context);

470 void
471 OpcGenerateAmlOpcode (
472     ACPI_PARSE_OBJECT       *Op);

474 UINT32
475 OpcSetOptimalIntegerSize (
476     ACPI_PARSE_OBJECT       *Op);

478 void
479 OpcGetIntegerWidth (
480     ACPI_PARSE_OBJECT       *Op);


483 /*
484  * asloperands - generate AML operands for the AML opcodes
485  */
486 ACPI_PARSE_OBJECT  *
487 UtGetArg (
488     ACPI_PARSE_OBJECT       *Op,
489     UINT32                  Argn);

491 void
492 OpnGenerateAmlOperands (
493     ACPI_PARSE_OBJECT       *Op);

495 void
496 OpnDoPackage (
497     ACPI_PARSE_OBJECT       *Op);


500 /*
501  * aslopt - optmization
502  */
503 void
504 OptOptimizeNamePath (
505     ACPI_PARSE_OBJECT       *Op,
506     UINT32                  Flags,
507     ACPI_WALK_STATE         *WalkState,
508     char                    *AmlNameString,
509     ACPI_NAMESPACE_NODE     *TargetNode);


512 /*
513  * aslcodegen - code generation
514  */
515 void
516 CgGenerateAmlOutput (
517     void);


520 /*
521  * aslfile
```

```
 */
void
FlOpenFile (
    UINT32                  FileId,
    char                    *Filename,
    char                    *Mode);


/*
 * asllength - calculate/adjust AML package lengths
 */
ACPI_STATUS
LnPackageLengthWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

ACPI_STATUS
LnInitLengthsWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

void
CgGenerateAmlLengths (
    ACPI_PARSE_OBJECT       *Op);


/*
 * aslmap - opcode mappings and reserved method names
 */
ACPI_OBJECT_TYPE
AslMapNamedOpcodeToDataType (
    UINT16                  Opcode);


/*
 * aslpredef - ACPI predefined names support
 */
BOOLEAN
ApCheckForPredefinedMethod (
    ACPI_PARSE_OBJECT       *Op,
    ASL_METHOD_INFO         *MethodInfo);

void
ApCheckPredefinedReturnValue (
    ACPI_PARSE_OBJECT       *Op,
    ASL_METHOD_INFO         *MethodInfo);

UINT32
ApCheckForPredefinedName (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name);

void
ApCheckForPredefinedObject (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name);

ACPI_STATUS
ApCheckObjectType (
    const char              *PredefinedName,
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  ExpectedBtypes,
    UINT32                  PackageIndex);
```



```
void
ApDisplayReservedNames (
    void);


/*
 * aslprepkg - ACPI predefined names support for packages
 */
void
ApCheckPackage (
    ACPI_PARSE_OBJECT           *ParentOp,
    const ACPI_PREDEFINED_INFO  *Predefined);


/*
 * asltransform - parse tree transformations
 */
ACPI_STATUS
TrAmlTransformWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);


/*
 * asltree - parse tree support
 */
ACPI_STATUS
TrWalkParseTree (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Visitation,
    ASL_WALK_CALLBACK       DescendingCallback,
    ASL_WALK_CALLBACK       AscendingCallback,
    void                    *Context);

/* Values for "Visitation" parameter above */

#define ASL_WALK_VISIT_DOWNWARD     0x01
#define ASL_WALK_VISIT_UPWARD       0x02
#define ASL_WALK_VISIT_TWICE        (ASL_WALK_VISIT_DOWNWARD | ASL_WALK_VISIT_UP


ACPI_PARSE_OBJECT *
TrAllocateNode (
    UINT32                  ParseOpcode);

void
TrReleaseNode (
    ACPI_PARSE_OBJECT       *Op);

ACPI_PARSE_OBJECT *
TrUpdateNode (
    UINT32                  ParseOpcode,
    ACPI_PARSE_OBJECT       *Op);

ACPI_PARSE_OBJECT *
TrCreateNode (
    UINT32                  ParseOpcode,
    UINT32                  NumChildren,
    ...);

ACPI_PARSE_OBJECT *
TrCreateLeafNode (
    UINT32                  ParseOpcode);

ACPI_PARSE_OBJECT *
```

```
654 TrCreateValuedLeafNode (
655     UINT32                  ParseOpcode,
656     UINT64                  Value);

658 ACPI_PARSE_OBJECT *
659 TrCreateConstantLeafNode (
660     UINT32                  ParseOpcode);

662 ACPI_PARSE_OBJECT *
663 TrLinkChildren (
664     ACPI_PARSE_OBJECT       *Op,
665     UINT32                  NumChildren,
666     ...);

668 void
669 TrSetEndLineNumber (
670     ACPI_PARSE_OBJECT       *Op);

672 void
673 TrWalkTree (
674     void);

676 ACPI_PARSE_OBJECT *
677 TrLinkPeerNode (
678     ACPI_PARSE_OBJECT       *Op1,
679     ACPI_PARSE_OBJECT       *Op2);

681 ACPI_PARSE_OBJECT *
682 TrLinkChildNode (
683     ACPI_PARSE_OBJECT       *Op1,
684     ACPI_PARSE_OBJECT       *Op2);

686 ACPI_PARSE_OBJECT *
687 TrSetNodeFlags (
688     ACPI_PARSE_OBJECT       *Op,
689     UINT32                  Flags);

691 ACPI_PARSE_OBJECT *
692 TrSetNodeAmlLength (
693     ACPI_PARSE_OBJECT       *Op,
694     UINT32                  Length);

696 ACPI_PARSE_OBJECT *
697 TrLinkPeerNodes (
698     UINT32                  NumPeers,
699     ...);


702 /*
703  * aslfiles - File I/O support
704  */
705 void
706 AslAbort (
707     void);

709 void
710 FlAddIncludeDirectory (
711     char                    *Dir);

713 char *
714 FlMergePathnames (
715     char                    *PrefixDir,
716     char                    *FilePathname);

718 void
719 FlOpenIncludeFile (
```



```
720     ACPI_PARSE_OBJECT       *Op);

722 void
723 FlFileError (
724     UINT32                  FileId,
725     UINT8                   ErrorId);

727 UINT32
728 FlGetFileSize (
729     UINT32                  FileId);

731 ACPI_STATUS
732 FlReadFile (
733     UINT32                  FileId,
734     void                    *Buffer,
735     UINT32                  Length);

737 void
738 FlWriteFile (
739     UINT32                  FileId,
740     void                    *Buffer,
741     UINT32                  Length);

743 void
744 FlSeekFile (
745     UINT32                  FileId,
746     long                    Offset);

748 void
749 FlCloseFile (
750     UINT32                  FileId);

752 void
753 FlPrintFile (
754     UINT32                  FileId,
755     char                    *Format,
756     ...);

758 void
759 FlDeleteFile (
760     UINT32                  FileId);

762 void
763 FlSetLineNumber (
764     UINT32                  LineNumber);

766 void
767 FlSetFilename (
768     char                    *Filename);

770 ACPI_STATUS
771 FlOpenInputFile (
772     char                    *InputFilename);

774 ACPI_STATUS
775 FlOpenAmlOutputFile (
776     char                    *InputFilename);

778 ACPI_STATUS
779 FlOpenMiscOutputFiles (
780     char                    *InputFilename);


783 /*
784  * asload - load namespace in prep for cross reference
785  */
```

```
786 ACPI_STATUS
787 LdLoadNamespace (
788     ACPI_PARSE_OBJECT       *RootOp);


791 /*
792  * asllookup - namespace lookup functions
793  */
794 void
795 LkFindUnreferencedObjects (
796     void);

798 /*
799  * aslmain - startup
800  */
801 void
802 Usage (
803     void);

805 void
806 AslFilenameHelp (
807     void);


810 /*
811  * aslnamesp - namespace output file generation
812  */
813 ACPI_STATUS
814 NsDisplayNamespace (
815     void);

817 void
818 NsSetupNamespaceListing (
819     void                    *Handle);

821 /*
822  * asloptions - command line processing
823  */
824 int
825 AslCommandLine (
826     int                     argc,
827     char                    **argv);

829 /*
830  * aslxref - namespace cross reference
831  */
832 ACPI_STATUS
833 XfCrossReferenceNamespace (
834     void);


837 /*
838  * aslutils - common compiler utilites
839  */
840 void
841 DbgPrint (
842     UINT32                  Type,
843     char                    *Format,
844     ...);

846 /* Type values for above */

848 #define ASL_DEBUG_OUTPUT    0
849 #define ASL_PARSE_OUTPUT    1
850 #define ASL_TREE_OUTPUT     2
```



```
852 void
853 UtDisplaySupportedTables (
854     void);

856 void
857 UtDisplayConstantOpcodes (
858     void);

860 UINT8
861 UtBeginEvent (
862     char                    *Name);

864 void
865 UtEndEvent (
866     UINT8                   Event);

868 void *
869 UtLocalCalloc (
870     UINT32                  Size);

872 void
873 UtPrintFormattedName (
874     UINT16                  ParseOpcode,
875     UINT32                  Level);

877 void
878 UtDisplaySummary (
879     UINT32                  FileId);

881 UINT8
882 UtHexCharToValue (
883     int                     HexChar);

885 void
886 UtConvertByteToHex (
887     UINT8                   RawByte,
888     UINT8                   *Buffer);

890 void
891 UtConvertByteToAsmHex (
892     UINT8                   RawByte,
893     UINT8                   *Buffer);

895 char *
896 UtGetOpName (
897     UINT32                  ParseOpcode);

899 void
900 UtSetParseOpName (
901     ACPI_PARSE_OBJECT       *Op);

903 char *
904 UtGetStringBuffer (
905     UINT32                  Length);

907 void
908 UtExpandLineBuffers (
909     void);

911 ACPI_STATUS
912 UtInternalizeName (
913     char                    *ExternalName,
914     char                    **ConvertedName);

916 void
917 UtAttachNamepathToOwner (
```

```
    ACPI_PARSE_OBJECT       *Op,
    ACPI_PARSE_OBJECT       *NameNode);

ACPI_PARSE_OBJECT *
UtCheckIntegerRange (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  LowValue,
    UINT32                  HighValue);

UINT64
UtDoConstant (
    char                    *String);

ACPI_STATUS
UtStrtoul64 (
    char                    *String,
    UINT32                  Base,
    UINT64                  *RetInteger);


/*
 * asluuid - UUID support
 */
ACPI_STATUS
AuValidateUuid (
    char                    *InString);

ACPI_STATUS
AuConvertStringToUuid (
    char                    *InString,
    char                    *UuIdBuffer);

ACPI_STATUS
AuConvertUuidToString (
    char                    *UuIdBuffer,
    char                    *OutString);

/*
 * aslresource - Resource template generation utilities
 */
void
RsSmallAddressCheck (
    UINT8                   Type,
    UINT32                  Minimum,
    UINT32                  Maximum,
    UINT32                  Length,
    UINT32                  Alignment,
    ACPI_PARSE_OBJECT       *MinOp,
    ACPI_PARSE_OBJECT       *MaxOp,
    ACPI_PARSE_OBJECT       *LengthOp,
    ACPI_PARSE_OBJECT       *AlignOp,
    ACPI_PARSE_OBJECT       *Op);

void
RsLargeAddressCheck (
    UINT64                  Minimum,
    UINT64                  Maximum,
    UINT64                  Length,
    UINT64                  Granularity,
    UINT8                   Flags,
    ACPI_PARSE_OBJECT       *MinOp,
    ACPI_PARSE_OBJECT       *MaxOp,
    ACPI_PARSE_OBJECT       *LengthOp,
    ACPI_PARSE_OBJECT       *GranOp,
    ACPI_PARSE_OBJECT       *Op);
```



```
UINT16
RsGetStringDataLength (
    ACPI_PARSE_OBJECT       *InitializerOp);

ASL_RESOURCE_NODE *
RsAllocateResourceNode (
    UINT32                  Size);

void
RsCreateResourceField (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name,
    UINT32                  ByteOffset,
    UINT32                  BitOffset,
    UINT32                  BitLength);

void
RsSetFlagBits (
    UINT8                   *Flags,
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   Position,
    UINT8                   DefaultBit);

void
RsSetFlagBits16 (
    UINT16                  *Flags,
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   Position,
    UINT8                   DefaultBit);

ACPI_PARSE_OBJECT *
RsCompleteNodeAndGetNext (
    ACPI_PARSE_OBJECT       *Op);

void
RsCheckListForDuplicates (
    ACPI_PARSE_OBJECT       *Op);

ASL_RESOURCE_NODE *
RsDoOneResourceDescriptor (
    ACPI_PARSE_OBJECT       *DescriptorTypeOp,
    UINT32                  CurrentByteOffset,
    UINT8                   *State);

/* Values for State above */

#define ACPI_RSTATE_NORMAL              0
#define ACPI_RSTATE_START_DEPENDENT     1
#define ACPI_RSTATE_DEPENDENT_LIST      2

UINT32
RsLinkDescriptorChain (
    ASL_RESOURCE_NODE       **PreviousRnode,
    ASL_RESOURCE_NODE       *Rnode);

void
RsDoResourceTemplate (
    ACPI_PARSE_OBJECT       *Op);


/*
 * aslrestype1 - Miscellaneous Small descriptors
 */
ASL_RESOURCE_NODE *
RsDoEndTagDescriptor (
    ACPI_PARSE_OBJECT       *Op,
```

```
1050     UINT32                  CurrentByteOffset);

1052 ASL_RESOURCE_NODE *
1053 RsDoEndDependentDescriptor (
1054     ACPI_PARSE_OBJECT       *Op,
1055     UINT32                  CurrentByteOffset);

1057 ASL_RESOURCE_NODE *
1058 RsDoMemory24Descriptor (
1059     ACPI_PARSE_OBJECT       *Op,
1060     UINT32                  CurrentByteOffset);

1062 ASL_RESOURCE_NODE *
1063 RsDoMemory32Descriptor (
1064     ACPI_PARSE_OBJECT       *Op,
1065     UINT32                  CurrentByteOffset);

1067 ASL_RESOURCE_NODE *
1068 RsDoMemory32FixedDescriptor (
1069     ACPI_PARSE_OBJECT       *Op,
1070     UINT32                  CurrentByteOffset);

1072 ASL_RESOURCE_NODE *
1073 RsDoStartDependentDescriptor (
1074     ACPI_PARSE_OBJECT       *Op,
1075     UINT32                  CurrentByteOffset);

1077 ASL_RESOURCE_NODE *
1078 RsDoStartDependentNoPriDescriptor (
1079     ACPI_PARSE_OBJECT       *Op,
1080     UINT32                  CurrentByteOffset);

1082 ASL_RESOURCE_NODE *
1083 RsDoVendorSmallDescriptor (
1084     ACPI_PARSE_OBJECT       *Op,
1085     UINT32                  CurrentByteOffset);


1088 /*
1089  * aslrestype1i - I/O-related Small descriptors
1090  */
1091 ASL_RESOURCE_NODE *
1092 RsDoDmaDescriptor (
1093     ACPI_PARSE_OBJECT       *Op,
1094     UINT32                  CurrentByteOffset);

1096 ASL_RESOURCE_NODE *
1097 RsDoFixedDmaDescriptor (
1098     ACPI_PARSE_OBJECT       *Op,
1099     UINT32                  CurrentByteOffset);

1101 ASL_RESOURCE_NODE *
1102 RsDoFixedIoDescriptor (
1103     ACPI_PARSE_OBJECT       *Op,
1104     UINT32                  CurrentByteOffset);

1106 ASL_RESOURCE_NODE *
1107 RsDoIoDescriptor (
1108     ACPI_PARSE_OBJECT       *Op,
1109     UINT32                  CurrentByteOffset);

1111 ASL_RESOURCE_NODE *
1112 RsDoIrqDescriptor (
1113     ACPI_PARSE_OBJECT       *Op,
1114     UINT32                  CurrentByteOffset);
```



```
1116 ASL_RESOURCE_NODE *
1117 RsDoIrqNoFlagsDescriptor (
1118     ACPI_PARSE_OBJECT       *Op,
1119     UINT32                  CurrentByteOffset);


1122 /*
1123  * aslrestype2 - Large resource descriptors
1124  */
1125 ASL_RESOURCE_NODE *
1126 RsDoInterruptDescriptor (
1127     ACPI_PARSE_OBJECT       *Op,
1128     UINT32                  CurrentByteOffset);

1130 ASL_RESOURCE_NODE *
1131 RsDoVendorLargeDescriptor (
1132     ACPI_PARSE_OBJECT       *Op,
1133     UINT32                  CurrentByteOffset);

1135 ASL_RESOURCE_NODE *
1136 RsDoGeneralRegisterDescriptor (
1137     ACPI_PARSE_OBJECT       *Op,
1138     UINT32                  CurrentByteOffset);

1140 ASL_RESOURCE_NODE *
1141 RsDoGpioIntDescriptor (
1142     ACPI_PARSE_OBJECT       *Op,
1143     UINT32                  CurrentByteOffset);

1145 ASL_RESOURCE_NODE *
1146 RsDoGpioIoDescriptor (
1147     ACPI_PARSE_OBJECT       *Op,
1148     UINT32                  CurrentByteOffset);

1150 ASL_RESOURCE_NODE *
1151 RsDoI2cSerialBusDescriptor (
1152     ACPI_PARSE_OBJECT       *Op,
1153     UINT32                  CurrentByteOffset);

1155 ASL_RESOURCE_NODE *
1156 RsDoSpiSerialBusDescriptor (
1157     ACPI_PARSE_OBJECT       *Op,
1158     UINT32                  CurrentByteOffset);

1160 ASL_RESOURCE_NODE *
1161 RsDoUartSerialBusDescriptor (
1162     ACPI_PARSE_OBJECT       *Op,
1163     UINT32                  CurrentByteOffset);

1165 /*
1166  * aslrestype2d - DWord address descriptors
1167  */
1168 ASL_RESOURCE_NODE *
1169 RsDoDwordIoDescriptor (
1170     ACPI_PARSE_OBJECT       *Op,
1171     UINT32                  CurrentByteOffset);

1173 ASL_RESOURCE_NODE *
1174 RsDoDwordMemoryDescriptor (
1175     ACPI_PARSE_OBJECT       *Op,
1176     UINT32                  CurrentByteOffset);

1178 ASL_RESOURCE_NODE *
1179 RsDoDwordSpaceDescriptor (
1180     ACPI_PARSE_OBJECT       *Op,
1181     UINT32                  CurrentByteOffset);
```

```
1184 /*
1185  * aslrestype2e - Extended address descriptors
1186  */
1187 ASL_RESOURCE_NODE *
1188 RsDoExtendedIoDescriptor (
1189     ACPI_PARSE_OBJECT       *Op,
1190     UINT32                  CurrentByteOffset);

1192 ASL_RESOURCE_NODE *
1193 RsDoExtendedMemoryDescriptor (
1194     ACPI_PARSE_OBJECT       *Op,
1195     UINT32                  CurrentByteOffset);

1197 ASL_RESOURCE_NODE *
1198 RsDoExtendedSpaceDescriptor (
1199     ACPI_PARSE_OBJECT       *Op,
1200     UINT32                  CurrentByteOffset);


1203 /*
1204  * aslrestype2q - QWord address descriptors
1205  */
1206 ASL_RESOURCE_NODE *
1207 RsDoQwordIoDescriptor (
1208     ACPI_PARSE_OBJECT       *Op,
1209     UINT32                  CurrentByteOffset);

1211 ASL_RESOURCE_NODE *
1212 RsDoQwordMemoryDescriptor (
1213     ACPI_PARSE_OBJECT       *Op,
1214     UINT32                  CurrentByteOffset);

1216 ASL_RESOURCE_NODE *
1217 RsDoQwordSpaceDescriptor (
1218     ACPI_PARSE_OBJECT       *Op,
1219     UINT32                  CurrentByteOffset);


1222 /*
1223  * aslrestype2w - Word address descriptors
1224  */
1225 ASL_RESOURCE_NODE *
1226 RsDoWordIoDescriptor (
1227     ACPI_PARSE_OBJECT       *Op,
1228     UINT32                  CurrentByteOffset);

1230 ASL_RESOURCE_NODE *
1231 RsDoWordSpaceDescriptor (
1232     ACPI_PARSE_OBJECT       *Op,
1233     UINT32                  CurrentByteOffset);

1235 ASL_RESOURCE_NODE *
1236 RsDoWordBusNumberDescriptor (
1237     ACPI_PARSE_OBJECT       *Op,
1238     UINT32                  CurrentByteOffset);

1240 /*
1241  * Entry to data table compiler subsystem
1242  */
1243 ACPI_STATUS
1244 DtDoCompile(
1245     void);

1247 ACPI_STATUS
```



```
1248 DtCreateTemplates (
1249     char                    *Signature);

1251 #endif /*  __ASLCOMPILER_H */
```

```
**********************************************************
   33668 Thu Dec 26 13:48:26 2013
new/usr/src/common/acpica/compiler/aslcompiler.l
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: aslcompiler.l - Flex/lex input file
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "aslcompiler.y.h"

#include <stdlib.h>
#include <string.h>
YYSTYPE AslCompilerlval;

/*
 * Generation:  Use the following command line:
 *
 * flex.exe -PAslCompiler -i -o$(InputPath).c $(InputPath)
 *
 * -i: Scanner must be case-insensitive
 */

#define _COMPONENT          ACPI_COMPILER
```



```
        ACPI_MODULE_NAME    ("aslscanner")


/* Local prototypes */

static void
AslDoLineDirective (void);

static char
AslDoComment (void);

static char
AslDoCommentType2 (void);

static char
AslDoStringLiteral (void);

static void
count (int type);


/*! [Begin] no source code translation */

%}
    /* Definitions */

LeadNameChar                [A-Za-z_]
DigitChar                   [0-9]
HexDigitChar                [A-Fa-f0-9]
RootChar                    [\\]
Nothing                     []

NameChar                    [A-Za-z_0-9]
NameSeg1                    {LeadNameChar}{NameChar}
NameSeg2                    {LeadNameChar}{NameChar}{NameChar}
NameSeg3                    {LeadNameChar}{NameChar}{NameChar}{NameChar}
NameSeg                     {LeadNameChar}|{NameSeg1}|{NameSeg2}|{NameSeg3}

NameString                  {RootChar}|{RootChar}{NamePath}|[\^]+{NamePath}|{Non
NamePath                    {NonEmptyNamePath}?
NonEmptyNamePath            {NameSeg}{NamePathTail}*
NamePathTail                [.]{NameSeg}

%%
    /* Rules */

[ ]                         { count (0); }
[\n]                        { count (0); } /* Handle files with both LF and CR/L
[\r]                        { count (0); } /* termination on both Unix and Windo
[ \t]                       { count (0); }


"/*"                        { if (!AslDoComment ()) yyterminate (); }
"//"                        { if (!AslDoCommentType2 ()) yyterminate (); }

"\""                        { if (AslDoStringLiteral ()) return (PARSEOP_STRING_
";"                         { count (0); return(';'); }


0[xX]{HexDigitChar}+ |
{DigitChar}+                { AslCompilerlval.i = UtDoConstant ((char *) AslComp
                                count (1); return (PARSEOP_INTEGER); }

"Include"                   { count (1); return (PARSEOP_INCLUDE); }
"External"                  { count (1); return (PARSEOP_EXTERNAL); }
```

```
 /*
  * The #line directive is emitted by the preprocessor and handled
  * here in the main iASL lexer - simply set the line number and
  * optionally the current filename.
  */
"#line"                     { AslDoLineDirective ();}


 /******************************************************************************
  *
  * Main ASL operators
  *
  *****************************************************************************/

"AccessAs"                  { count (1); return (PARSEOP_ACCESSAS); }
"Acquire"                   { count (3); return (PARSEOP_ACQUIRE); }
"Add"                       { count (3); return (PARSEOP_ADD); }
"Alias"                     { count (2); return (PARSEOP_ALIAS); }
"And"                       { count (3); return (PARSEOP_AND); }
"BankField"                 { count (2); return (PARSEOP_BANKFIELD); }
"Break"                     { count (3); return (PARSEOP_BREAK); }
"BreakPoint"                { count (3); return (PARSEOP_BREAKPOINT); }
"Buffer"                    { count (1); return (PARSEOP_BUFFER); }
"Case"                      { count (3); return (PARSEOP_CASE); }
"Concatenate"               { count (3); return (PARSEOP_CONCATENATE); }
"ConcatenateResTemplate"    { count (3); return (PARSEOP_CONCATENATERESTEMPLATE)
"CondRefOf"                 { count (3); return (PARSEOP_CONDREFOF); }
"Connection"                { count (2); return (PARSEOP_CONNECTION); }
"Continue"                  { count (3); return (PARSEOP_CONTINUE); }
"CopyObject"                { count (3); return (PARSEOP_COPYOBJECT); }
"CreateBitField"            { count (2); return (PARSEOP_CREATEBITFIELD); }
"CreateByteField"           { count (2); return (PARSEOP_CREATEBYTEFIELD); }
"CreateDWordField"          { count (2); return (PARSEOP_CREATEDWORDFIELD); }
"CreateField"               { count (2); return (PARSEOP_CREATEFIELD); }
"CreateQWordField"          { count (2); return (PARSEOP_CREATEQWORDFIELD); }
"CreateWordField"           { count (2); return (PARSEOP_CREATEWORDFIELD); }
"DataTableRegion"           { count (2); return (PARSEOP_DATATABLEREGION); }
"Debug"                     { count (1); return (PARSEOP_DEBUG); }
"Decrement"                 { count (3); return (PARSEOP_DECREMENT); }
"Default"                   { count (3); return (PARSEOP_DEFAULT); }
"DefinitionBlock"           { count (1); return (PARSEOP_DEFINITIONBLOCK); }
"DeRefOf"                   { count (3); return (PARSEOP_DEREFOF); }
"Device"                    { count (2); return (PARSEOP_DEVICE); }
"Divide"                    { count (3); return (PARSEOP_DIVIDE); }
"Eisaid"                    { count (1); return (PARSEOP_EISAID); }
"Else"                      { count (3); return (PARSEOP_ELSE); }
"ElseIf"                    { count (3); return (PARSEOP_ELSEIF); }
"Event"                     { count (2); return (PARSEOP_EVENT); }
"Fatal"                     { count (3); return (PARSEOP_FATAL); }
"Field"                     { count (2); return (PARSEOP_FIELD); }
"FindSetLeftBit"            { count (3); return (PARSEOP_FINDSETLEFTBIT); }
"FindSetRightBit"           { count (3); return (PARSEOP_FINDSETRIGHTBIT); }
"FromBcd"                   { count (3); return (PARSEOP_FROMBCD); }
"Function"                  { count (2); return (PARSEOP_FUNCTION); }
"If"                        { count (3); return (PARSEOP_IF); }
"Increment"                 { count (3); return (PARSEOP_INCREMENT); }
"Index"                     { count (3); return (PARSEOP_INDEX); }
"IndexField"                { count (2); return (PARSEOP_INDEXFIELD); }
"LAnd"                      { count (3); return (PARSEOP_LAND); }
"LEqual"                    { count (3); return (PARSEOP_LEQUAL); }
"LGreater"                  { count (3); return (PARSEOP_LGREATER); }
"LGreaterEqual"             { count (3); return (PARSEOP_LGREATEREQUAL); }
"LLess"                     { count (3); return (PARSEOP_LLESS); }
"LLessEqual"                { count (3); return (PARSEOP_LLESSEQUAL); }
"LNot"                      { count (3); return (PARSEOP_LNOT); }
"LNotEqual"                 { count (3); return (PARSEOP_LNOTEQUAL); }
```



```
"Load"                      { count (3); return (PARSEOP_LOAD); }
"LoadTable"                 { count (3); return (PARSEOP_LOADTABLE); }
"LOr"                       { count (3); return (PARSEOP_LOR); }
"Match"                     { count (3); return (PARSEOP_MATCH); }
"Method"                    { count (2); return (PARSEOP_METHOD); }
"Mid"                       { count (3); return (PARSEOP_MID); }
"Mod"                       { count (3); return (PARSEOP_MOD); }
"Multiply"                  { count (3); return (PARSEOP_MULTIPLY); }
"Mutex"                     { count (2); return (PARSEOP_MUTEX); }
"Name"                      { count (2); return (PARSEOP_NAME); }
"NAnd"                      { count (3); return (PARSEOP_NAND); }
"Noop"                      { if (!AcpiGbl_IgnoreNoopOperator) {count (3); retur
"NOr"                       { count (3); return (PARSEOP_NOR); }
"Not"                       { count (3); return (PARSEOP_NOT); }
"Notify"                    { count (3); return (PARSEOP_NOTIFY); }
"ObjectType"                { count (3); return (PARSEOP_OBJECTTYPE); }
"Offset"                    { count (1); return (PARSEOP_OFFSET); }
"One"                       { count (1); return (PARSEOP_ONE); }
"Ones"                      { count (1); return (PARSEOP_ONES); }
"OperationRegion"           { count (2); return (PARSEOP_OPERATIONREGION); }
"Or"                        { count (3); return (PARSEOP_OR); }
"Package"                   { count (1); return (PARSEOP_PACKAGE); }
"PowerResource"             { count (2); return (PARSEOP_POWERRESOURCE); }
"Processor"                 { count (2); return (PARSEOP_PROCESSOR); }
"RefOf"                     { count (3); return (PARSEOP_REFOF); }
"Release"                   { count (3); return (PARSEOP_RELEASE); }
"Reset"                     { count (3); return (PARSEOP_RESET); }
"Return"                    { count (3); return (PARSEOP_RETURN); }
"Revision"                  { count (1); return (PARSEOP_REVISION); }
"Scope"                     { count (2); return (PARSEOP_SCOPE); }
"ShiftLeft"                 { count (3); return (PARSEOP_SHIFTLEFT); }
"ShiftRight"                { count (3); return (PARSEOP_SHIFTRIGHT); }
"Signal"                    { count (3); return (PARSEOP_SIGNAL); }
"SizeOf"                    { count (3); return (PARSEOP_SIZEOF); }
"Sleep"                     { count (3); return (PARSEOP_SLEEP); }
"Stall"                     { count (3); return (PARSEOP_STALL); }
"Store"                     { count (3); return (PARSEOP_STORE); }
"Subtract"                  { count (3); return (PARSEOP_SUBTRACT); }
"Switch"                    { count (3); return (PARSEOP_SWITCH); }
"ThermalZone"               { count (2); return (PARSEOP_THERMALZONE); }
"Timer"                     { count (3); return (PARSEOP_TIMER); }
"ToBcd"                     { count (3); return (PARSEOP_TOBCD); }
"ToBuffer"                  { count (3); return (PARSEOP_TOBUFFER); }
"ToDecimalString"           { count (3); return (PARSEOP_TODECIMALSTRING); }
"ToHexString"               { count (3); return (PARSEOP_TOHEXSTRING); }
"ToInteger"                 { count (3); return (PARSEOP_TOINTEGER); }
"ToString"                  { count (3); return (PARSEOP_TOSTRING); }
"ToUuid"                    { count (1); return (PARSEOP_TOUUID); }
"Unicode"                   { count (1); return (PARSEOP_UNICODE); }
"Unload"                    { count (3); return (PARSEOP_UNLOAD); }
"Wait"                      { count (3); return (PARSEOP_WAIT); }
"While"                     { count (3); return (PARSEOP_WHILE); }
"XOr"                       { count (3); return (PARSEOP_XOR); }
"Zero"                      { count (1); return (PARSEOP_ZERO); }

    /* Control method arguments and locals */

"Arg0"                      { count (1); return (PARSEOP_ARG0); }
"Arg1"                      { count (1); return (PARSEOP_ARG1); }
"Arg2"                      { count (1); return (PARSEOP_ARG2); }
"Arg3"                      { count (1); return (PARSEOP_ARG3); }
"Arg4"                      { count (1); return (PARSEOP_ARG4); }
"Arg5"                      { count (1); return (PARSEOP_ARG5); }
"Arg6"                      { count (1); return (PARSEOP_ARG6); }
"Local0"                    { count (1); return (PARSEOP_LOCAL0); }
"Local1"                    { count (1); return (PARSEOP_LOCAL1); }
```

```
"Local2"                { count (1); return (PARSEOP_LOCAL2); }
"Local3"                { count (1); return (PARSEOP_LOCAL3); }
"Local4"                { count (1); return (PARSEOP_LOCAL4); }
"Local5"                { count (1); return (PARSEOP_LOCAL5); }
"Local6"                { count (1); return (PARSEOP_LOCAL6); }
"Local7"                { count (1); return (PARSEOP_LOCAL7); }


 /*******************************************************************************
  *
  * Resource Descriptor macros
  *
  ******************************************************************************/

"ResourceTemplate"          { count (1); return (PARSEOP_RESOURCETEMPLATE); }
"RawDataBuffer"             { count (1); return (PARSEOP_DATABUFFER); }

"DMA"                       { count (1); return (PARSEOP_DMA); }
"DWordIO"                   { count (1); return (PARSEOP_DWORDIO); }
"DWordMemory"               { count (1); return (PARSEOP_DWORDMEMORY); }
"DWordSpace"                { count (1); return (PARSEOP_DWORDSPACE); }
"EndDependentFn"            { count (1); return (PARSEOP_ENDDEPENDENTFN); }
"ExtendedIO"                { count (1); return (PARSEOP_EXTENDEDIO); }
"ExtendedMemory"            { count (1); return (PARSEOP_EXTENDEDMEMORY); }
"ExtendedSpace"             { count (1); return (PARSEOP_EXTENDEDSPACE); }
"FixedDma"                  { count (1); return (PARSEOP_FIXEDDMA); }
"FixedIO"                   { count (1); return (PARSEOP_FIXEDIO); }
"GpioInt"                   { count (1); return (PARSEOP_GPIO_INT); }
"GpioIo"                    { count (1); return (PARSEOP_GPIO_IO); }
"I2cSerialBus"              { count (1); return (PARSEOP_I2C_SERIALBUS); }
"Interrupt"                 { count (1); return (PARSEOP_INTERRUPT); }
"IO"                        { count (1); return (PARSEOP_IO); }
"IRQ"                       { count (1); return (PARSEOP_IRQ); }
"IRQNoFlags"                { count (1); return (PARSEOP_IRQNOFLAGS); }
"Memory24"                  { count (1); return (PARSEOP_MEMORY24); }
"Memory32"                  { count (1); return (PARSEOP_MEMORY32); }
"Memory32Fixed"             { count (1); return (PARSEOP_MEMORY32FIXED); }
"QWordIO"                   { count (1); return (PARSEOP_QWORDIO); }
"QWordMemory"               { count (1); return (PARSEOP_QWORDMEMORY); }
"QWordSpace"                { count (1); return (PARSEOP_QWORDSPACE); }
"Register"                  { count (1); return (PARSEOP_REGISTER); }
"SpiSerialBus"              { count (1); return (PARSEOP_SPI_SERIALBUS); }
"StartDependentFn"          { count (1); return (PARSEOP_STARTDEPENDENTFN); }
"StartDependentFnNoPri"     { count (1); return (PARSEOP_STARTDEPENDENTFN_NOPRI)
"UartSerialBus"             { count (1); return (PARSEOP_UART_SERIALBUS); }
"VendorLong"                { count (1); return (PARSEOP_VENDORLONG); }
"VendorShort"               { count (1); return (PARSEOP_VENDORSHORT); }
"WordBusNumber"             { count (1); return (PARSEOP_WORDBUSNUMBER); }
"WordIO"                    { count (1); return (PARSEOP_WORDIO); }
"WordSpace"                 { count (1); return (PARSEOP_WORDSPACE); }


 /*******************************************************************************
  *
  * Keywords used as arguments to ASL operators and macros
  *
  ******************************************************************************/

    /*  AccessAttribKeyword: Serial Bus Attributes (ACPI 5.0) */

"AttribQuick"               { count (0); return (PARSEOP_ACCESSATTRIB_QUICK); }
"AttribSendReceive"         { count (0); return (PARSEOP_ACCESSATTRIB_SND_RCV);
"AttribByte"                { count (0); return (PARSEOP_ACCESSATTRIB_BYTE); }
"AttribWord"                { count (0); return (PARSEOP_ACCESSATTRIB_WORD); }
"AttribBlock"               { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK); }
"AttribProcessCall"         { count (0); return (PARSEOP_ACCESSATTRIB_WORD_CALL)
```



```
"AttribBlockProcessCall"    { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK_CALL

    /* AccessAttribKeyword: Legacy synonyms for above (pre-ACPI 5.0) */

"SMBQuick"                  { count (0); return (PARSEOP_ACCESSATTRIB_QUICK); }
"SMBSendReceive"            { count (0); return (PARSEOP_ACCESSATTRIB_SND_RCV);
"SMBByte"                   { count (0); return (PARSEOP_ACCESSATTRIB_BYTE); }
"SMBWord"                   { count (0); return (PARSEOP_ACCESSATTRIB_WORD); }
"SMBBlock"                  { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK); }
"SMBProcessCall"            { count (0); return (PARSEOP_ACCESSATTRIB_WORD_CALL)
"SMBBlockProcessCall"       { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK_CALL

    /* AccessTypeKeyword: Field Access Types */

"AnyAcc"                    { count (0); return (PARSEOP_ACCESSTYPE_ANY); }
"ByteAcc"                   { count (0); return (PARSEOP_ACCESSTYPE_BYTE); }
"WordAcc"                   { count (0); return (PARSEOP_ACCESSTYPE_WORD); }
"DWordAcc"                  { count (0); return (PARSEOP_ACCESSTYPE_DWORD); }
"QWordAcc"                  { count (0); return (PARSEOP_ACCESSTYPE_QWORD); }
"BufferAcc"                 { count (0); return (PARSEOP_ACCESSTYPE_BUF); }

    /* AddressingModeKeyword: Mode - Resource Descriptors (ACPI 5.0) */

"AddressingMode7Bit"        { count (0); return (PARSEOP_ADDRESSINGMODE_7BIT); }
"AddressingMode10Bit"       { count (0); return (PARSEOP_ADDRESSINGMODE_10BIT);

    /* AddressKeyword: ACPI memory range types */

"AddressRangeMemory"        { count (0); return (PARSEOP_ADDRESSTYPE_MEMORY); }
"AddressRangeReserved"      { count (0); return (PARSEOP_ADDRESSTYPE_RESERVED);
"AddressRangeNVS"           { count (0); return (PARSEOP_ADDRESSTYPE_NVS); }
"AddressRangeACPI"          { count (0); return (PARSEOP_ADDRESSTYPE_ACPI); }

    /* BusMasterKeyword: DMA Bus Mastering */

"BusMaster"                 { count (0); return (PARSEOP_BUSMASTERTYPE_MASTER);
"NotBusMaster"              { count (0); return (PARSEOP_BUSMASTERTYPE_NOTMASTER

    /* ByteLengthKeyword: Bits per Byte - Resource Descriptors (ACPI 5.0) */

"DataBitsFive"              { count (0); return (PARSEOP_BITSPERBYTE_FIVE); }
"DataBitsSix"               { count (0); return (PARSEOP_BITSPERBYTE_SIX); }
"DataBitsSeven"             { count (0); return (PARSEOP_BITSPERBYTE_SEVEN); }
"DataBitsEight"             { count (0); return (PARSEOP_BITSPERBYTE_EIGHT); }
"DataBitsNine"              { count (0); return (PARSEOP_BITSPERBYTE_NINE); }

    /* ClockPhaseKeyword: Resource Descriptors (ACPI 5.0) */

"ClockPhaseFirst"           { count (0); return (PARSEOP_CLOCKPHASE_FIRST); }
"ClockPhaseSecond"          { count (0); return (PARSEOP_CLOCKPHASE_SECOND); }

    /* ClockPolarityKeyword: Resource Descriptors (ACPI 5.0) */

"ClockPolarityLow"          { count (0); return (PARSEOP_CLOCKPOLARITY_LOW); }
"ClockPolarityHigh"         { count (0); return (PARSEOP_CLOCKPOLARITY_HIGH); }

    /* DecodeKeyword: Type of Memory Decoding - Resource Descriptors */

"PosDecode"                 { count (0); return (PARSEOP_DECODETYPE_POS); }
"SubDecode"                 { count (0); return (PARSEOP_DECODETYPE_SUB); }

    /* DmaTypeKeyword: DMA Types - DMA Resource Descriptor */

"Compatibility"             { count (0); return (PARSEOP_DMATYPE_COMPATIBILITY);
"TypeA"                     { count (0); return (PARSEOP_DMATYPE_A); }
"TypeB"                     { count (0); return (PARSEOP_DMATYPE_B); }
```

```
391 "TypeF"                     { count (0); return (PARSEOP_DMATYPE_F); }

393     /* EndianKeyword: Endian type - Resource Descriptor (ACPI 5.0) */

395 "LittleEndian"              { count (0); return (PARSEOP_ENDIAN_LITTLE); }
396 "BigEndian"                 { count (0); return (PARSEOP_ENDIAN_BIG); }

398     /* ExtendedAttribKeyword: Bus attributes, AccessAs operator (ACPI 5.0) */

400 "AttribBytes"               { count (0); return (PARSEOP_ACCESSATTRIB_MULTIBYTE)
401 "AttribRawBytes"            { count (0); return (PARSEOP_ACCESSATTRIB_RAW_BYTES)
402 "AttribRawProcessBytes"     { count (0); return (PARSEOP_ACCESSATTRIB_RAW_PROCES

404     /* FlowControlKeyword: Resource Descriptors (ACPI 5.0) */

406 "FlowControlHardware"       { count (0); return (PARSEOP_FLOWCONTROL_HW); }
407 "FlowControlNone"           { count (0); return (PARSEOP_FLOWCONTROL_NONE); }
408 "FlowControlXon"            { count (0); return (PARSEOP_FLOWCONTROL_SW); }

410     /* InterruptLevelKeyword: Interrupt Active Types */

412 "ActiveBoth"                { count (0); return (PARSEOP_INTLEVEL_ACTIVEBOTH); }
413 "ActiveHigh"                { count (0); return (PARSEOP_INTLEVEL_ACTIVEHIGH); }
414 "ActiveLow"                 { count (0); return (PARSEOP_INTLEVEL_ACTIVELOW); }

416     /* InterruptTypeKeyword: Interrupt Types */

418 "Edge"                      { count (0); return (PARSEOP_INTTYPE_EDGE); }
419 "Level"                     { count (0); return (PARSEOP_INTTYPE_LEVEL); }

421     /* IoDecodeKeyword: Type of Memory Decoding - Resource Descriptors */

423 "Decode10"                  { count (0); return (PARSEOP_IODECODETYPE_10); }
424 "Decode16"                  { count (0); return (PARSEOP_IODECODETYPE_16); }

426     /* IoRestrictionKeyword: I/O Restriction - GPIO Resource Descriptors (ACPI 5

428 "IoRestrictionNone"         { count (0); return (PARSEOP_IORESTRICT_NONE); }
429 "IoRestrictionInputOnly"    { count (0); return (PARSEOP_IORESTRICT_IN); }
430 "IoRestrictionOutputOnly"   { count (0); return (PARSEOP_IORESTRICT_OUT); }
431 "IoRestrictionNoneAndPreserve"   { count (0); return (PARSEOP_IORESTRICT_PRESERV

433     /* LockRuleKeyword: Global Lock use for Field Operator */

435 "Lock"                      { count (0); return (PARSEOP_LOCKRULE_LOCK); }
436 "NoLock"                    { count (0); return (PARSEOP_LOCKRULE_NOLOCK); }

438     /* MatchOpKeyword: Types for Match Operator */

440 "MTR"                       { count (0); return (PARSEOP_MATCHTYPE_MTR); }
441 "MEQ"                       { count (0); return (PARSEOP_MATCHTYPE_MEQ); }
442 "MLE"                       { count (0); return (PARSEOP_MATCHTYPE_MLE); }
443 "MLT"                       { count (0); return (PARSEOP_MATCHTYPE_MLT); }
444 "MGE"                       { count (0); return (PARSEOP_MATCHTYPE_MGE); }
445 "MGT"                       { count (0); return (PARSEOP_MATCHTYPE_MGT); }

447     /* MaxKeyword: Max Range Type - Resource Descriptors */

449 "MaxFixed"                  { count (0); return (PARSEOP_MAXTYPE_FIXED); }
450 "MaxNotFixed"               { count (0); return (PARSEOP_MAXTYPE_NOTFIXED); }

452     /* MemTypeKeyword: Memory Types - Resource Descriptors */

454 "Cacheable"                 { count (0); return (PARSEOP_MEMTYPE_CACHEABLE); }
455 "WriteCombining"            { count (0); return (PARSEOP_MEMTYPE_WRITECOMBINING)
456 "Prefetchable"              { count (0); return (PARSEOP_MEMTYPE_PREFETCHABLE);
```



```
457 "NonCacheable"             { count (0); return (PARSEOP_MEMTYPE_NONCACHEABLE);

459     /* MinKeyword: Min Range Type - Resource Descriptors */

461 "MinFixed"                  { count (0); return (PARSEOP_MINTYPE_FIXED); }
462 "MinNotFixed"               { count (0); return (PARSEOP_MINTYPE_NOTFIXED); }

464     /* ObjectTypeKeyword: ACPI Object Types */

466 "UnknownObj"                { count (0); return (PARSEOP_OBJECTTYPE_UNK); }
467 "IntObj"                    { count (0); return (PARSEOP_OBJECTTYPE_INT); }
468 "StrObj"                    { count (0); return (PARSEOP_OBJECTTYPE_STR); }
469 "BuffObj"                   { count (0); return (PARSEOP_OBJECTTYPE_BUF); }
470 "PkgObj"                    { count (0); return (PARSEOP_OBJECTTYPE_PKG); }
471 "FieldUnitObj"              { count (0); return (PARSEOP_OBJECTTYPE_FLD); }
472 "DeviceObj"                 { count (0); return (PARSEOP_OBJECTTYPE_DEV); }
473 "EventObj"                  { count (0); return (PARSEOP_OBJECTTYPE_EVT); }
474 "MethodObj"                 { count (0); return (PARSEOP_OBJECTTYPE_MTH); }
475 "MutexObj"                  { count (0); return (PARSEOP_OBJECTTYPE_MTX); }
476 "OpRegionObj"               { count (0); return (PARSEOP_OBJECTTYPE_OPR); }
477 "PowerResObj"               { count (0); return (PARSEOP_OBJECTTYPE_POW); }
478 "ProcessorObj"              { count (0); return (PARSEOP_OBJECTTYPE_PRO); }
479 "ThermalZoneObj"            { count (0); return (PARSEOP_OBJECTTYPE_THZ); }
480 "BuffFieldObj"              { count (0); return (PARSEOP_OBJECTTYPE_BFF); }
481 "DDBHandleObj"              { count (0); return (PARSEOP_OBJECTTYPE_DDB); }

483     /* ParityKeyword: Resource Descriptors (ACPI 5.0) */

485 "ParityTypeSpace"           { count (0); return (PARSEOP_PARITYTYPE_SPACE); }
486 "ParityTypeMark"            { count (0); return (PARSEOP_PARITYTYPE_MARK); }
487 "ParityTypeOdd"             { count (0); return (PARSEOP_PARITYTYPE_ODD); }
488 "ParityTypeEven"            { count (0); return (PARSEOP_PARITYTYPE_EVEN); }
489 "ParityTypeNone"            { count (0); return (PARSEOP_PARITYTYPE_NONE); }

491     /* PinConfigKeyword: Pin Configuration - GPIO Resource Descriptors (ACPI 5.0

493 "PullDefault"               { count (0); return (PARSEOP_PIN_PULLDEFAULT); }
494 "PullUp"                    { count (0); return (PARSEOP_PIN_PULLUP); }
495 "PullDown"                  { count (0); return (PARSEOP_PIN_PULLDOWN); }
496 "PullNone"                  { count (0); return (PARSEOP_PIN_NOPULL); }

498     /* PolarityKeyword: Resource Descriptors (ACPI 5.0) */

500 "PolarityLow"               { count (0); return (PARSEOP_DEVICEPOLARITY_LOW); }
501 "PolarityHigh"              { count (0); return (PARSEOP_DEVICEPOLARITY_HIGH); }

503     /* RangeTypeKeyword: I/O Range Types - Resource Descriptors */

505 "ISAOnlyRanges"             { count (0); return (PARSEOP_RANGETYPE_ISAONLY); }
506 "NonISAOnlyRanges"          { count (0); return (PARSEOP_RANGETYPE_NONISAONLY);
507 "EntireRange"               { count (0); return (PARSEOP_RANGETYPE_ENTIRE); }

509     /* ReadWriteKeyword: Memory Access Types - Resource Descriptors */

511 "ReadWrite"                 { count (0); return (PARSEOP_READWRITETYPE_BOTH); }
512 "ReadOnly"                  { count (0); return (PARSEOP_READWRITETYPE_READONLY)

514     /* RegionSpaceKeyword: Operation Region Address Space Types */

516 "SystemIO"                  { count (0); return (PARSEOP_REGIONSPACE_IO); }
517 "SystemMemory"              { count (0); return (PARSEOP_REGIONSPACE_MEM); }
518 "PCI_Config"                { count (0); return (PARSEOP_REGIONSPACE_PCI); }
519 "EmbeddedControl"           { count (0); return (PARSEOP_REGIONSPACE_EC); }
520 "SMBus"                     { count (0); return (PARSEOP_REGIONSPACE_SMBUS); }
521 "SystemCMOS"                { count (0); return (PARSEOP_REGIONSPACE_CMOS); }
522 "PciBarTarget"              { count (0); return (PARSEOP_REGIONSPACE_PCIBAR); }
```

```
"IPMI"                  { count (0); return (PARSEOP_REGIONSPACE_IPMI); }
"GeneralPurposeIo"      { count (0); return (PARSEOP_REGIONSPACE_GPIO); }
"GenericSerialBus"      { count (0); return (PARSEOP_REGIONSPACE_GSBUS); }
"PCC"                   { count (0); return (PARSEOP_REGIONSPACE_PCC); }
"FFixedHW"              { count (0); return (PARSEOP_REGIONSPACE_FFIXEDHW);

    /* ResourceTypeKeyword: Resource Usage - Resource Descriptors */

"ResourceConsumer"      { count (0); return (PARSEOP_RESOURCETYPE_CONSUMER);
"ResourceProducer"      { count (0); return (PARSEOP_RESOURCETYPE_PRODUCER);

    /* SerializeRuleKeyword: Control Method Serialization */

"Serialized"            { count (0); return (PARSEOP_SERIALIZERULE_SERIAL);
"NotSerialized"         { count (0); return (PARSEOP_SERIALIZERULE_NOTSERIAL

    /* ShareTypeKeyword: Interrupt Sharing - Resource Descriptors */

"Shared"                { count (0); return (PARSEOP_SHARETYPE_SHARED); }
"Exclusive"             { count (0); return (PARSEOP_SHARETYPE_EXCLUSIVE); }
"SharedAndWake"         { count (0); return (PARSEOP_SHARETYPE_SHAREDWAKE);
"ExclusiveAndWake"      { count (0); return (PARSEOP_SHARETYPE_EXCLUSIVEWAKE

    /* SlaveModeKeyword: Resource Descriptors (ACPI 5.0) */

"ControllerInitiated"   { count (0); return (PARSEOP_SLAVEMODE_CONTROLLERINI
"DeviceInitiated"       { count (0); return (PARSEOP_SLAVEMODE_DEVICEINIT);

    /* StopBitsKeyword: Resource Descriptors (ACPI 5.0) */

"StopBitsOne"           { count (0); return (PARSEOP_STOPBITS_ONE); }
"StopBitsOnePlusHalf"   { count (0); return (PARSEOP_STOPBITS_ONEPLUSHALF);
"StopBitsTwo"           { count (0); return (PARSEOP_STOPBITS_TWO); }
"StopBitsZero"          { count (0); return (PARSEOP_STOPBITS_ZERO); }

    /* TransferWidthKeyword: DMA Widths - Fixed DMA Resource Descriptor (ACPI 5.

"Width8bit"             { count (0); return (PARSEOP_XFERSIZE_8); }
"Width16bit"            { count (0); return (PARSEOP_XFERSIZE_16); }
"Width32bit"            { count (0); return (PARSEOP_XFERSIZE_32); }
"Width64bit"            { count (0); return (PARSEOP_XFERSIZE_64); }
"Width128bit"           { count (0); return (PARSEOP_XFERSIZE_128); }
"Width256bit"           { count (0); return (PARSEOP_XFERSIZE_256); }

    /* TranslationKeyword: Translation Density Types - Resource Descriptors */

"SparseTranslation"     { count (0); return (PARSEOP_TRANSLATIONTYPE_SPARSE)
"DenseTranslation"      { count (0); return (PARSEOP_TRANSLATIONTYPE_DENSE);

    /* TypeKeyword: Translation Types - Resource Descriptors */

"TypeTranslation"       { count (0); return (PARSEOP_TYPE_TRANSLATION); }
"TypeStatic"            { count (0); return (PARSEOP_TYPE_STATIC); }

    /* UpdateRuleKeyword: Field Update Rules */

"Preserve"              { count (0); return (PARSEOP_UPDATERULE_PRESERVE); }
"WriteAsOnes"           { count (0); return (PARSEOP_UPDATERULE_ONES); }
"WriteAsZeros"          { count (0); return (PARSEOP_UPDATERULE_ZEROS); }

    /* WireModeKeyword: SPI Wire Mode - Resource Descriptors (ACPI 5.0) */

"FourWireMode"          { count (0); return (PARSEOP_WIREMODE_FOUR); }
"ThreeWireMode"         { count (0); return (PARSEOP_WIREMODE_THREE); }

    /* XferTypeKeyword: DMA Transfer Types */
```



```
"Transfer8"             { count (0); return (PARSEOP_XFERTYPE_8); }
"Transfer8_16"          { count (0); return (PARSEOP_XFERTYPE_8_16); }
"Transfer16"            { count (0); return (PARSEOP_XFERTYPE_16); }

    /* Predefined compiler names */

"__DATE__"              { count (0); return (PARSEOP___DATE__); }
"__FILE__"              { count (0); return (PARSEOP___FILE__); }
"__LINE__"              { count (0); return (PARSEOP___LINE__); }
"__PATH__"              { count (0); return (PARSEOP___PATH__); }


"{"                     { count (0); return('{'); }
"}"                     { count (0); return('}'); }
","                     { count (0); return(','); }
"("                     { count (0); return('('); }
")"                     { count (0); return(')'); }

{NameSeg}               { char *s;
                          count (0);
                          s=malloc (ACPI_NAME_SIZE + 1);
                          if (strcmp (AslCompilertext, "\\"))
                          {
                              strcpy (s, "____");
                              AcpiUtStrupr (AslCompilertext);
                          }
                          memcpy (s, AslCompilertext, strlen (AslCompilert
                          AslCompilerlval.s = s;
                          DbgPrint (ASL_PARSE_OUTPUT, "NameSeg: %s\n", s);
                          return (PARSEOP_NAMESEG); }

{NameString}            { char *s;
                          count (0);
                          s=malloc (strlen (AslCompilertext)+1);
                          AcpiUtStrupr (AslCompilertext);
                          strcpy (s, AslCompilertext);
                          s[strlen (AslCompilertext)] = 0;
                          AslCompilerlval.s = s;
                          DbgPrint (ASL_PARSE_OUTPUT, "NameString: %s\n",
                          return (PARSEOP_NAMESTRING); }

"*" |
"/"                     { count (1);
                          AslCompilererror ("Parse error, expecting ASL ke

.                       { count (1);
                          sprintf (MsgBuffer,
                              "Invalid character (0x%2.2X), expecting ASL
                              *AslCompilertext);
                          AslCompilererror (MsgBuffer);}

<<EOF>>                 { if (AslPopInputFileStack ())
                            yyterminate();
                          else
                            return (PARSEOP_INCLUDE_END);};

%%

/*! [End] no source code translation !*/

/*
 * Bring in the scanner support routines
 */
#include "aslsupport.l"
```

```
**********************************************************
  126239 Thu Dec 26 13:48:26 2013
new/usr/src/common/acpica/compiler/aslcompiler.y
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: aslcompiler.y - Bison/Yacc input file (ASL grammar and actions)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslparse")

/*
 * Global Notes:
 *
 * October 2005: The following list terms have been optimized (from the
 * original ASL grammar in the ACPI specification) to force the immediate
 * reduction of each list item so that the parse stack use doesn't increase on
```



```
 * each list element and possibly overflow on very large lists (>4000 items).
 * This dramatically reduces use of the parse stack overall.
 *
 *      ArgList, TermList, Objectlist, ByteList, DWordList, PackageList,
 *      ResourceMacroList, and FieldUnitList
 */

void *                      AslLocalAllocate (unsigned int Size);

/* Bison/yacc configuration */

#define static
#undef alloca
#define alloca              AslLocalAllocate
#define yytname             AslCompilername

#define YYINITDEPTH         600             /* State stack depth */
#define YYDEBUG             1               /* Enable debug output */
#define YYERROR_VERBOSE     1               /* Verbose error messages */

/* Define YYMALLOC/YYFREE to prevent redefinition errors  */

#define YYMALLOC            malloc
#define YYFREE              free

/*
 * The windows version of bison defines this incorrectly as "32768" (Not negativ
 * We use a custom (edited binary) version of bison that defines YYFLAG as YYFBA
 * instead (#define YYFBAD 32768), so we can define it correctly here.
 *
 * The problem is that if YYFLAG is positive, the extended syntax error messages
 * are disabled.
 */
#define YYFLAG              -32768

%}

/*
 * Declare the type of values in the grammar
 */
%union {
    UINT64              i;
    char                *s;
    ACPI_PARSE_OBJECT   *n;
}

/*! [Begin] no source code translation */

/*
 * These shift/reduce conflicts are expected. There should be zero
 * reduce/reduce conflicts.
 */
%expect 86

/******************************************************************************
 *
 * Token types: These are returned by the lexer
 *
 * NOTE: This list MUST match the AslKeywordMapping table found
 *       in aslmap.c EXACTLY!  Double check any changes!
 *
 *****************************************************************************/

%token <i> PARSEOP_ACCESSAS
%token <i> PARSEOP_ACCESSATTRIB_BLOCK
%token <i> PARSEOP_ACCESSATTRIB_BLOCK_CALL
```

```
%token <i> PARSEOP_ACCESSATTRIB_BYTE
%token <i> PARSEOP_ACCESSATTRIB_MULTIBYTE
%token <i> PARSEOP_ACCESSATTRIB_QUICK
%token <i> PARSEOP_ACCESSATTRIB_RAW_BYTES
%token <i> PARSEOP_ACCESSATTRIB_RAW_PROCESS
%token <i> PARSEOP_ACCESSATTRIB_SND_RCV
%token <i> PARSEOP_ACCESSATTRIB_WORD
%token <i> PARSEOP_ACCESSATTRIB_WORD_CALL
%token <i> PARSEOP_ACCESSTYPE_ANY
%token <i> PARSEOP_ACCESSTYPE_BUF
%token <i> PARSEOP_ACCESSTYPE_BYTE
%token <i> PARSEOP_ACCESSTYPE_DWORD
%token <i> PARSEOP_ACCESSTYPE_QWORD
%token <i> PARSEOP_ACCESSTYPE_WORD
%token <i> PARSEOP_ACQUIRE
%token <i> PARSEOP_ADD
%token <i> PARSEOP_ADDRESSINGMODE_7BIT
%token <i> PARSEOP_ADDRESSINGMODE_10BIT
%token <i> PARSEOP_ADDRESSTYPE_ACPI
%token <i> PARSEOP_ADDRESSTYPE_MEMORY
%token <i> PARSEOP_ADDRESSTYPE_NVS
%token <i> PARSEOP_ADDRESSTYPE_RESERVED
%token <i> PARSEOP_ALIAS
%token <i> PARSEOP_AND
%token <i> PARSEOP_ARG0
%token <i> PARSEOP_ARG1
%token <i> PARSEOP_ARG2
%token <i> PARSEOP_ARG3
%token <i> PARSEOP_ARG4
%token <i> PARSEOP_ARG5
%token <i> PARSEOP_ARG6
%token <i> PARSEOP_BANKFIELD
%token <i> PARSEOP_BITSPERBYTE_EIGHT
%token <i> PARSEOP_BITSPERBYTE_FIVE
%token <i> PARSEOP_BITSPERBYTE_NINE
%token <i> PARSEOP_BITSPERBYTE_SEVEN
%token <i> PARSEOP_BITSPERBYTE_SIX
%token <i> PARSEOP_BREAK
%token <i> PARSEOP_BREAKPOINT
%token <i> PARSEOP_BUFFER
%token <i> PARSEOP_BUSMASTERTYPE_MASTER
%token <i> PARSEOP_BUSMASTERTYPE_NOTMASTER
%token <i> PARSEOP_BYTECONST
%token <i> PARSEOP_CASE
%token <i> PARSEOP_CLOCKPHASE_FIRST
%token <i> PARSEOP_CLOCKPHASE_SECOND
%token <i> PARSEOP_CLOCKPOLARITY_HIGH
%token <i> PARSEOP_CLOCKPOLARITY_LOW
%token <i> PARSEOP_CONCATENATE
%token <i> PARSEOP_CONCATENATERESTEMPLATE
%token <i> PARSEOP_CONDREFOF
%token <i> PARSEOP_CONNECTION
%token <i> PARSEOP_CONTINUE
%token <i> PARSEOP_COPYOBJECT
%token <i> PARSEOP_CREATEBITFIELD
%token <i> PARSEOP_CREATEBYTEFIELD
%token <i> PARSEOP_CREATEDWORDFIELD
%token <i> PARSEOP_CREATEFIELD
%token <i> PARSEOP_CREATEQWORDFIELD
%token <i> PARSEOP_CREATEWORDFIELD
%token <i> PARSEOP_DATABUFFER
%token <i> PARSEOP_DATATABLEREGION
%token <i> PARSEOP_DEBUG
%token <i> PARSEOP_DECODETYPE_POS
%token <i> PARSEOP_DECODETYPE_SUB
%token <i> PARSEOP_DECREMENT
```



```
%token <i> PARSEOP_DEFAULT
%token <i> PARSEOP_DEFAULT_ARG
%token <i> PARSEOP_DEFINITIONBLOCK
%token <i> PARSEOP_DEREFOF
%token <i> PARSEOP_DEVICE
%token <i> PARSEOP_DEVICEPOLARITY_HIGH
%token <i> PARSEOP_DEVICEPOLARITY_LOW
%token <i> PARSEOP_DIVIDE
%token <i> PARSEOP_DMA
%token <i> PARSEOP_DMATYPE_A
%token <i> PARSEOP_DMATYPE_COMPATIBILITY
%token <i> PARSEOP_DMATYPE_B
%token <i> PARSEOP_DMATYPE_F
%token <i> PARSEOP_DWORDCONST
%token <i> PARSEOP_DWORDIO
%token <i> PARSEOP_DWORDMEMORY
%token <i> PARSEOP_DWORDSPACE
%token <i> PARSEOP_EISAID
%token <i> PARSEOP_ELSE
%token <i> PARSEOP_ELSEIF
%token <i> PARSEOP_ENDDEPENDENTFN
%token <i> PARSEOP_ENDIAN_BIG
%token <i> PARSEOP_ENDIAN_LITTLE
%token <i> PARSEOP_ENDTAG
%token <i> PARSEOP_ERRORNODE
%token <i> PARSEOP_EVENT
%token <i> PARSEOP_EXTENDEDIO
%token <i> PARSEOP_EXTENDEDMEMORY
%token <i> PARSEOP_EXTENDEDSPACE
%token <i> PARSEOP_EXTERNAL
%token <i> PARSEOP_FATAL
%token <i> PARSEOP_FIELD
%token <i> PARSEOP_FINDSETLEFTBIT
%token <i> PARSEOP_FINDSETRIGHTBIT
%token <i> PARSEOP_FIXEDDMA
%token <i> PARSEOP_FIXEDIO
%token <i> PARSEOP_FLOWCONTROL_HW
%token <i> PARSEOP_FLOWCONTROL_NONE
%token <i> PARSEOP_FLOWCONTROL_SW
%token <i> PARSEOP_FROMBCD
%token <i> PARSEOP_FUNCTION
%token <i> PARSEOP_GPIO_INT
%token <i> PARSEOP_GPIO_IO
%token <i> PARSEOP_I2C_SERIALBUS
%token <i> PARSEOP_IF
%token <i> PARSEOP_INCLUDE
%token <i> PARSEOP_INCLUDE_END
%token <i> PARSEOP_INCREMENT
%token <i> PARSEOP_INDEX
%token <i> PARSEOP_INDEXFIELD
%token <i> PARSEOP_INTEGER
%token <i> PARSEOP_INTERRUPT
%token <i> PARSEOP_INTLEVEL_ACTIVEBOTH
%token <i> PARSEOP_INTLEVEL_ACTIVEHIGH
%token <i> PARSEOP_INTLEVEL_ACTIVELOW
%token <i> PARSEOP_INTTYPE_EDGE
%token <i> PARSEOP_INTTYPE_LEVEL
%token <i> PARSEOP_IO
%token <i> PARSEOP_IODECODETYPE_10
%token <i> PARSEOP_IODECODETYPE_16
%token <i> PARSEOP_IORESTRICT_IN
%token <i> PARSEOP_IORESTRICT_NONE
%token <i> PARSEOP_IORESTRICT_OUT
%token <i> PARSEOP_IORESTRICT_PRESERVE
%token <i> PARSEOP_IRQ
%token <i> PARSEOP_IRQNOFLAGS
```

```
%token <i> PARSEOP_LAND
%token <i> PARSEOP_LEQUAL
%token <i> PARSEOP_LGREATER
%token <i> PARSEOP_LGREATEREQUAL
%token <i> PARSEOP_LLESS
%token <i> PARSEOP_LLESSEQUAL
%token <i> PARSEOP_LNOT
%token <i> PARSEOP_LNOTEQUAL
%token <i> PARSEOP_LOAD
%token <i> PARSEOP_LOADTABLE
%token <i> PARSEOP_LOCAL0
%token <i> PARSEOP_LOCAL1
%token <i> PARSEOP_LOCAL2
%token <i> PARSEOP_LOCAL3
%token <i> PARSEOP_LOCAL4
%token <i> PARSEOP_LOCAL5
%token <i> PARSEOP_LOCAL6
%token <i> PARSEOP_LOCAL7
%token <i> PARSEOP_LOCKRULE_LOCK
%token <i> PARSEOP_LOCKRULE_NOLOCK
%token <i> PARSEOP_LOR
%token <i> PARSEOP_MATCH
%token <i> PARSEOP_MATCHTYPE_MEQ
%token <i> PARSEOP_MATCHTYPE_MGE
%token <i> PARSEOP_MATCHTYPE_MGT
%token <i> PARSEOP_MATCHTYPE_MLE
%token <i> PARSEOP_MATCHTYPE_MLT
%token <i> PARSEOP_MATCHTYPE_MTR
%token <i> PARSEOP_MAXTYPE_FIXED
%token <i> PARSEOP_MAXTYPE_NOTFIXED
%token <i> PARSEOP_MEMORY24
%token <i> PARSEOP_MEMORY32
%token <i> PARSEOP_MEMORY32FIXED
%token <i> PARSEOP_MEMTYPE_CACHEABLE
%token <i> PARSEOP_MEMTYPE_NONCACHEABLE
%token <i> PARSEOP_MEMTYPE_PREFETCHABLE
%token <i> PARSEOP_MEMTYPE_WRITECOMBINING
%token <i> PARSEOP_METHOD
%token <i> PARSEOP_METHODCALL
%token <i> PARSEOP_MID
%token <i> PARSEOP_MINTYPE_FIXED
%token <i> PARSEOP_MINTYPE_NOTFIXED
%token <i> PARSEOP_MOD
%token <i> PARSEOP_MULTIPLY
%token <i> PARSEOP_MUTEX
%token <i> PARSEOP_NAME
%token <s> PARSEOP_NAMESEG
%token <s> PARSEOP_NAMESTRING
%token <i> PARSEOP_NAND
%token <i> PARSEOP_NOOP
%token <i> PARSEOP_NOR
%token <i> PARSEOP_NOT
%token <i> PARSEOP_NOTIFY
%token <i> PARSEOP_OBJECTTYPE
%token <i> PARSEOP_OBJECTTYPE_BFF
%token <i> PARSEOP_OBJECTTYPE_BUF
%token <i> PARSEOP_OBJECTTYPE_DDB
%token <i> PARSEOP_OBJECTTYPE_DEV
%token <i> PARSEOP_OBJECTTYPE_EVT
%token <i> PARSEOP_OBJECTTYPE_FLD
%token <i> PARSEOP_OBJECTTYPE_INT
%token <i> PARSEOP_OBJECTTYPE_MTH
%token <i> PARSEOP_OBJECTTYPE_MTX
%token <i> PARSEOP_OBJECTTYPE_OPR
%token <i> PARSEOP_OBJECTTYPE_PKG
%token <i> PARSEOP_OBJECTTYPE_POW
```



```
%token <i> PARSEOP_OBJECTTYPE_PRO
%token <i> PARSEOP_OBJECTTYPE_STR
%token <i> PARSEOP_OBJECTTYPE_THZ
%token <i> PARSEOP_OBJECTTYPE_UNK
%token <i> PARSEOP_OFFSET
%token <i> PARSEOP_ONE
%token <i> PARSEOP_ONES
%token <i> PARSEOP_OPERATIONREGION
%token <i> PARSEOP_OR
%token <i> PARSEOP_PACKAGE
%token <i> PARSEOP_PACKAGE_LENGTH
%token <i> PARSEOP_PARITYTYPE_EVEN
%token <i> PARSEOP_PARITYTYPE_MARK
%token <i> PARSEOP_PARITYTYPE_NONE
%token <i> PARSEOP_PARITYTYPE_ODD
%token <i> PARSEOP_PARITYTYPE_SPACE
%token <i> PARSEOP_PIN_NOPULL
%token <i> PARSEOP_PIN_PULLDEFAULT
%token <i> PARSEOP_PIN_PULLDOWN
%token <i> PARSEOP_PIN_PULLUP
%token <i> PARSEOP_POWERRESOURCE
%token <i> PARSEOP_PROCESSOR
%token <i> PARSEOP_QWORDCONST
%token <i> PARSEOP_QWORDIO
%token <i> PARSEOP_QWORDMEMORY
%token <i> PARSEOP_QWORDSPACE
%token <i> PARSEOP_RANGETYPE_ENTIRE
%token <i> PARSEOP_RANGETYPE_ISAONLY
%token <i> PARSEOP_RANGETYPE_NONISAONLY
%token <i> PARSEOP_RAW_DATA
%token <i> PARSEOP_READWRITETYPE_BOTH
%token <i> PARSEOP_READWRITETYPE_READONLY
%token <i> PARSEOP_REFOF
%token <i> PARSEOP_REGIONSPACE_CMOS
%token <i> PARSEOP_REGIONSPACE_EC
%token <i> PARSEOP_REGIONSPACE_FFIXEDHW
%token <i> PARSEOP_REGIONSPACE_GPIO
%token <i> PARSEOP_REGIONSPACE_GSBUS
%token <i> PARSEOP_REGIONSPACE_IO
%token <i> PARSEOP_REGIONSPACE_IPMI
%token <i> PARSEOP_REGIONSPACE_MEM
%token <i> PARSEOP_REGIONSPACE_PCC
%token <i> PARSEOP_REGIONSPACE_PCI
%token <i> PARSEOP_REGIONSPACE_PCIBAR
%token <i> PARSEOP_REGIONSPACE_SMBUS
%token <i> PARSEOP_REGISTER
%token <i> PARSEOP_RELEASE
%token <i> PARSEOP_RESERVED_BYTES
%token <i> PARSEOP_RESET
%token <i> PARSEOP_RESOURCETEMPLATE
%token <i> PARSEOP_RESOURCETYPE_CONSUMER
%token <i> PARSEOP_RESOURCETYPE_PRODUCER
%token <i> PARSEOP_RETURN
%token <i> PARSEOP_REVISION
%token <i> PARSEOP_SCOPE
%token <i> PARSEOP_SERIALIZERULE_NOTSERIAL
%token <i> PARSEOP_SERIALIZERULE_SERIAL
%token <i> PARSEOP_SHARETYPE_EXCLUSIVE
%token <i> PARSEOP_SHARETYPE_EXCLUSIVEWAKE
%token <i> PARSEOP_SHARETYPE_SHARED
%token <i> PARSEOP_SHARETYPE_SHAREDWAKE
%token <i> PARSEOP_SHIFTLEFT
%token <i> PARSEOP_SHIFTRIGHT
%token <i> PARSEOP_SIGNAL
%token <i> PARSEOP_SIZEOF
%token <i> PARSEOP_SLAVEMODE_CONTROLLERINIT
```

```
391 %token <i> PARSEOP_SLAVEMODE_DEVICEINIT
392 %token <i> PARSEOP_SLEEP
393 %token <i> PARSEOP_SPI_SERIALBUS
394 %token <i> PARSEOP_STALL
395 %token <i> PARSEOP_STARTDEPENDENTFN
396 %token <i> PARSEOP_STARTDEPENDENTFN_NOPRI
397 %token <i> PARSEOP_STOPBITS_ONE
398 %token <i> PARSEOP_STOPBITS_ONEPLUSHALF
399 %token <i> PARSEOP_STOPBITS_TWO
400 %token <i> PARSEOP_STOPBITS_ZERO
401 %token <i> PARSEOP_STORE
402 %token <s> PARSEOP_STRING_LITERAL
403 %token <i> PARSEOP_SUBTRACT
404 %token <i> PARSEOP_SWITCH
405 %token <i> PARSEOP_THERMALZONE
406 %token <i> PARSEOP_TIMER
407 %token <i> PARSEOP_TOBCD
408 %token <i> PARSEOP_TOBUFFER
409 %token <i> PARSEOP_TODECIMALSTRING
410 %token <i> PARSEOP_TOHEXSTRING
411 %token <i> PARSEOP_TOINTEGER
412 %token <i> PARSEOP_TOSTRING
413 %token <i> PARSEOP_TOUUID
414 %token <i> PARSEOP_TRANSLATIONTYPE_DENSE
415 %token <i> PARSEOP_TRANSLATIONTYPE_SPARSE
416 %token <i> PARSEOP_TYPE_STATIC
417 %token <i> PARSEOP_TYPE_TRANSLATION
418 %token <i> PARSEOP_UART_SERIALBUS
419 %token <i> PARSEOP_UNICODE
420 %token <i> PARSEOP_UNLOAD
421 %token <i> PARSEOP_UPDATERULE_ONES
422 %token <i> PARSEOP_UPDATERULE_PRESERVE
423 %token <i> PARSEOP_UPDATERULE_ZEROS
424 %token <i> PARSEOP_VAR_PACKAGE
425 %token <i> PARSEOP_VENDORLONG
426 %token <i> PARSEOP_VENDORSHORT
427 %token <i> PARSEOP_WAIT
428 %token <i> PARSEOP_WHILE
429 %token <i> PARSEOP_WIREMODE_FOUR
430 %token <i> PARSEOP_WIREMODE_THREE
431 %token <i> PARSEOP_WORDBUSNUMBER
432 %token <i> PARSEOP_WORDCONST
433 %token <i> PARSEOP_WORDIO
434 %token <i> PARSEOP_WORDSPACE
435 %token <i> PARSEOP_XFERSIZE_8
436 %token <i> PARSEOP_XFERSIZE_16
437 %token <i> PARSEOP_XFERSIZE_32
438 %token <i> PARSEOP_XFERSIZE_64
439 %token <i> PARSEOP_XFERSIZE_128
440 %token <i> PARSEOP_XFERSIZE_256
441 %token <i> PARSEOP_XFERTYPE_8
442 %token <i> PARSEOP_XFERTYPE_8_16
443 %token <i> PARSEOP_XFERTYPE_16
444 %token <i> PARSEOP_XOR
445 %token <i> PARSEOP_ZERO

447 /*
448  * Special functions. These should probably stay at the end of this
449  * table.
450  */
451 %token <i> PARSEOP___DATE__
452 %token <i> PARSEOP___FILE__
453 %token <i> PARSEOP___LINE__
454 %token <i> PARSEOP___PATH__
```



```
457 /*******************************************************************************
458  *
459  * Production names
460  *
461  ******************************************************************************/

463 %type <n> ArgList
464 %type <n> ASLCode
465 %type <n> BufferData
466 %type <n> BufferTermData
467 %type <n> CompilerDirective
468 %type <n> DataObject
469 %type <n> DefinitionBlockTerm
470 %type <n> IntegerData
471 %type <n> NamedObject
472 %type <n> NameSpaceModifier
473 %type <n> Object
474 %type <n> ObjectList
475 %type <n> PackageData
476 %type <n> ParameterTypePackage
477 %type <n> ParameterTypePackageList
478 %type <n> ParameterTypesPackage
479 %type <n> ParameterTypesPackageList
480 %type <n> RequiredTarget
481 %type <n> SimpleTarget
482 %type <n> StringData
483 %type <n> Target
484 %type <n> Term
485 %type <n> TermArg
486 %type <n> TermList
487 %type <n> UserTerm

489 /* Type4Opcode is obsolete */

491 %type <n> Type1Opcode
492 %type <n> Type2BufferOpcode
493 %type <n> Type2BufferOrStringOpcode
494 %type <n> Type2IntegerOpcode
495 %type <n> Type2Opcode
496 %type <n> Type2StringOpcode
497 %type <n> Type3Opcode
498 %type <n> Type5Opcode
499 %type <n> Type6Opcode

501 %type <n> AccessAsTerm
502 %type <n> ExternalTerm
503 %type <n> FieldUnit
504 %type <n> FieldUnitEntry
505 %type <n> FieldUnitList
506 %type <n> IncludeTerm
507 %type <n> OffsetTerm
508 %type <n> OptionalAccessAttribTerm

510 /* Named Objects */

512 %type <n> BankFieldTerm
513 %type <n> CreateBitFieldTerm
514 %type <n> CreateByteFieldTerm
515 %type <n> CreateDWordFieldTerm
516 %type <n> CreateFieldTerm
517 %type <n> CreateQWordFieldTerm
518 %type <n> CreateWordFieldTerm
519 %type <n> DataRegionTerm
520 %type <n> DeviceTerm
521 %type <n> EventTerm
522 %type <n> FieldTerm
```

```
523 %type <n> FunctionTerm
524 %type <n> IndexFieldTerm
525 %type <n> MethodTerm
526 %type <n> MutexTerm
527 %type <n> OpRegionTerm
528 %type <n> OpRegionSpaceIdTerm
529 %type <n> PowerResTerm
530 %type <n> ProcessorTerm
531 %type <n> ThermalZoneTerm

533 /* Namespace modifiers */

535 %type <n> AliasTerm
536 %type <n> NameTerm
537 %type <n> ScopeTerm

539 /* Type 1 opcodes */

541 %type <n> BreakPointTerm
542 %type <n> BreakTerm
543 %type <n> CaseDefaultTermList
544 %type <n> CaseTerm
545 %type <n> ContinueTerm
546 %type <n> DefaultTerm
547 %type <n> ElseTerm
548 %type <n> FatalTerm
549 %type <n> IfElseTerm
550 %type <n> IfTerm
551 %type <n> LoadTerm
552 %type <n> NoOpTerm
553 %type <n> NotifyTerm
554 %type <n> ReleaseTerm
555 %type <n> ResetTerm
556 %type <n> ReturnTerm
557 %type <n> SignalTerm
558 %type <n> SleepTerm
559 %type <n> StallTerm
560 %type <n> SwitchTerm
561 %type <n> UnloadTerm
562 %type <n> WhileTerm
563 /* %type <n> CaseTermList */

565 /* Type 2 opcodes */

567 %type <n> AcquireTerm
568 %type <n> AddTerm
569 %type <n> AndTerm
570 %type <n> ConcatResTerm
571 %type <n> ConcatTerm
572 %type <n> CondRefOfTerm
573 %type <n> CopyObjectTerm
574 %type <n> DecTerm
575 %type <n> DerefOfTerm
576 %type <n> DivideTerm
577 %type <n> FindSetLeftBitTerm
578 %type <n> FindSetRightBitTerm
579 %type <n> FromBCDTerm
580 %type <n> IncTerm
581 %type <n> IndexTerm
582 %type <n> LAndTerm
583 %type <n> LEqualTerm
584 %type <n> LGreaterEqualTerm
585 %type <n> LGreaterTerm
586 %type <n> LLessEqualTerm
587 %type <n> LLessTerm
588 %type <n> LNotEqualTerm
```



```
589 %type <n> LNotTerm
590 %type <n> LoadTableTerm
591 %type <n> LOrTerm
592 %type <n> MatchTerm
593 %type <n> MidTerm
594 %type <n> ModTerm
595 %type <n> MultiplyTerm
596 %type <n> NAndTerm
597 %type <n> NOrTerm
598 %type <n> NotTerm
599 %type <n> ObjectTypeTerm
600 %type <n> OrTerm
601 %type <n> RefOfTerm
602 %type <n> ShiftLeftTerm
603 %type <n> ShiftRightTerm
604 %type <n> SizeOfTerm
605 %type <n> StoreTerm
606 %type <n> SubtractTerm
607 %type <n> TimerTerm
608 %type <n> ToBCDTerm
609 %type <n> ToBufferTerm
610 %type <n> ToDecimalStringTerm
611 %type <n> ToHexStringTerm
612 %type <n> ToIntegerTerm
613 %type <n> ToStringTerm
614 %type <n> WaitTerm
615 %type <n> XOrTerm

617 /* Keywords */

619 %type <n> AccessAttribKeyword
620 %type <n> AccessTypeKeyword
621 %type <n> AddressingModeKeyword
622 %type <n> AddressKeyword
623 %type <n> AddressSpaceKeyword
624 %type <n> BitsPerByteKeyword
625 %type <n> ClockPhaseKeyword
626 %type <n> ClockPolarityKeyword
627 %type <n> DecodeKeyword
628 %type <n> DevicePolarityKeyword
629 %type <n> DMATypeKeyword
630 %type <n> EndianKeyword
631 %type <n> FlowControlKeyword
632 %type <n> InterruptLevel
633 %type <n> InterruptTypeKeyword
634 %type <n> IODecodeKeyword
635 %type <n> IoRestrictionKeyword
636 %type <n> LockRuleKeyword
637 %type <n> MatchOpKeyword
638 %type <n> MaxKeyword
639 %type <n> MemTypeKeyword
640 %type <n> MinKeyword
641 %type <n> ObjectTypeKeyword
642 %type <n> OptionalBusMasterKeyword
643 %type <n> OptionalReadWriteKeyword
644 %type <n> ParityTypeKeyword
645 %type <n> PinConfigByte
646 %type <n> PinConfigKeyword
647 %type <n> RangeTypeKeyword
648 %type <n> RegionSpaceKeyword
649 %type <n> ResourceTypeKeyword
650 %type <n> SerializeRuleKeyword
651 %type <n> ShareTypeKeyword
652 %type <n> SlaveModeKeyword
653 %type <n> StopBitsKeyword
654 %type <n> TranslationKeyword
```

```
655 %type <n> TypeKeyword
656 %type <n> UpdateRuleKeyword
657 %type <n> WireModeKeyword
658 %type <n> XferSizeKeyword
659 %type <n> XferTypeKeyword

661 /* Types */

663 %type <n> SuperName
664 %type <n> ObjectTypeName
665 %type <n> ArgTerm
666 %type <n> LocalTerm
667 %type <n> DebugTerm

669 %type <n> Integer
670 %type <n> ByteConst
671 %type <n> WordConst
672 %type <n> DWordConst
673 %type <n> QWordConst
674 %type <n> String

676 %type <n> ConstTerm
677 %type <n> ConstExprTerm
678 %type <n> ByteConstExpr
679 %type <n> WordConstExpr
680 %type <n> DWordConstExpr
681 %type <n> QWordConstExpr

683 %type <n> DWordList
684 %type <n> BufferTerm
685 %type <n> ByteList

687 %type <n> PackageElement
688 %type <n> PackageList
689 %type <n> PackageTerm
690 %type <n> VarPackageLengthTerm

692 /* Macros */

694 %type <n> EISAIDTerm
695 %type <n> ResourceMacroList
696 %type <n> ResourceMacroTerm
697 %type <n> ResourceTemplateTerm
698 %type <n> ToUUIDTerm
699 %type <n> UnicodeTerm

701 /* Resource Descriptors */

703 %type <n> ConnectionTerm
704 %type <n> DataBufferTerm
705 %type <n> DMATerm
706 %type <n> DWordIOTerm
707 %type <n> DWordMemoryTerm
708 %type <n> DWordSpaceTerm
709 %type <n> EndDependentFnTerm
710 %type <n> ExtendedIOTerm
711 %type <n> ExtendedMemoryTerm
712 %type <n> ExtendedSpaceTerm
713 %type <n> FixedDmaTerm
714 %type <n> FixedIOTerm
715 %type <n> GpioIntTerm
716 %type <n> GpioIoTerm
717 %type <n> I2cSerialBusTerm
718 %type <n> InterruptTerm
719 %type <n> IOTerm
720 %type <n> IRQNoFlagsTerm
```



```
721 %type <n> IRQTerm
722 %type <n> Memory24Term
723 %type <n> Memory32FixedTerm
724 %type <n> Memory32Term
725 %type <n> NameSeg
726 %type <n> NameString
727 %type <n> QWordIOTerm
728 %type <n> QWordMemoryTerm
729 %type <n> QWordSpaceTerm
730 %type <n> RegisterTerm
731 %type <n> SpiSerialBusTerm
732 %type <n> StartDependentFnNoPriTerm
733 %type <n> StartDependentFnTerm
734 %type <n> UartSerialBusTerm
735 %type <n> VendorLongTerm
736 %type <n> VendorShortTerm
737 %type <n> WordBusNumberTerm
738 %type <n> WordIOTerm
739 %type <n> WordSpaceTerm

741 /* Local types that help construct the AML, not in ACPI spec */

743 %type <n> AmlPackageLengthTerm
744 %type <n> IncludeEndTerm
745 %type <n> NameStringItem
746 %type <n> TermArgItem

748 %type <n> OptionalAccessSize
749 %type <n> OptionalAddressingMode
750 %type <n> OptionalAddressRange
751 %type <n> OptionalBitsPerByte
752 %type <n> OptionalBuffer_Last
753 %type <n> OptionalByteConstExpr
754 %type <n> OptionalCount
755 %type <n> OptionalDecodeType
756 %type <n> OptionalDevicePolarity
757 %type <n> OptionalDWordConstExpr
758 %type <n> OptionalEndian
759 %type <n> OptionalFlowControl
760 %type <n> OptionalIoRestriction
761 %type <n> OptionalListString
762 %type <n> OptionalMaxType
763 %type <n> OptionalMemType
764 %type <n> OptionalMinType
765 %type <n> OptionalNameString
766 %type <n> OptionalNameString_First
767 %type <n> OptionalNameString_Last
768 %type <n> OptionalObjectTypeKeyword
769 %type <n> OptionalParameterTypePackage
770 %type <n> OptionalParameterTypesPackage
771 %type <n> OptionalParityType
772 %type <n> OptionalQWordConstExpr
773 %type <n> OptionalRangeType
774 %type <n> OptionalReference
775 %type <n> OptionalResourceType
776 %type <n> OptionalResourceType_First
777 %type <n> OptionalReturnArg
778 %type <n> OptionalSerializeRuleKeyword
779 %type <n> OptionalShareType
780 %type <n> OptionalShareType_First
781 %type <n> OptionalSlaveMode
782 %type <n> OptionalStopBits
783 %type <n> OptionalStringData
784 %type <n> OptionalTermArg
785 %type <n> OptionalTranslationType_Last
786 %type <n> OptionalType
```

```
%type <n> OptionalType_Last
%type <n> OptionalWireMode
%type <n> OptionalWordConst
%type <n> OptionalWordConstExpr
%type <n> OptionalXferSize

%%
/*******************************************************************************
 *
 * Production rules start here
 *
 ******************************************************************************/

/*
 * ASL Names
 */


/*
 * Root rule. Allow multiple #line directives before the definition block
 * to handle output from preprocessors
 */
ASLCode
    : DefinitionBlockTerm
    | error                         {YYABORT; $$ = NULL;}
    ;

/*
 * Blocks, Data, and Opcodes
 */

/*
 * Note concerning support for "module-level code".
 *
 * ACPI 1.0 allowed Type1 and Type2 executable opcodes outside of control
 * methods (the so-called module-level code.) This support was explicitly
 * removed in ACPI 2.0, but this type of code continues to be created by
 * BIOS vendors. In order to support the disassembly and recompilation of
 * such code (and the porting of ASL code to iASL), iASL supports this
 * code in violation of the current ACPI specification.
 *
 * The grammar change to support module-level code is to revert the
 * {ObjectList} portion of the DefinitionBlockTerm in ACPI 2.0 to the
 * original use of {TermList} instead (see below.) This allows the use
 * of Type1 and Type2 opcodes at module level.
 */
DefinitionBlockTerm
    : PARSEOP_DEFINITIONBLOCK '('   {$<n>$ = TrCreateLeafNode (PARSEOP_DEFINITIO
        String ','
        String ','
        ByteConst ','
        String ','
        String ','
        DWordConst
        ')'                         {TrSetEndLineNumber ($<n>3);}
            '{' TermList '}'        {$$ = TrLinkChildren ($<n>3,7,$4,$6,$8,$10,$
    ;

/* ACPI 3.0 -- allow semicolons between terms */

TermList
    :                               {$$ = NULL;}
    | TermList Term                 {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
    | TermList Term ';'             {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
    | TermList ';' Term             {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
    | TermList ';' Term ';'         {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
```



```
    ;

Term
    : Object                        {}
    | Type1Opcode                   {}
    | Type2Opcode                   {}
    | Type2IntegerOpcode            {}
    | Type2StringOpcode             {}
    | Type2BufferOpcode             {}
    | Type2BufferOrStringOpcode     {}
    | error                         {$$ = AslDoError(); yyclearin;}
    ;

CompilerDirective
    : IncludeTerm                   {}
    | ExternalTerm                  {}
    ;

ObjectList
    :                               {$$ = NULL;}
    | ObjectList Object             {$$ = TrLinkPeerNode ($1,$2);}
    | error                         {$$ = AslDoError(); yyclearin;}
    ;

Object
    : CompilerDirective             {}
    | NamedObject                   {}
    | NameSpaceModifier             {}
    ;

DataObject
    : BufferData                    {}
    | PackageData                   {}
    | IntegerData                   {}
    | StringData                    {}
    ;

BufferData
    : Type5Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | Type2BufferOrStringOpcode     {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | Type2BufferOpcode             {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | BufferTerm                    {}
    ;

PackageData
    : PackageTerm                   {}
    ;

IntegerData
    : Type2IntegerOpcode            {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | Type3Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | Integer                       {}
    | ConstTerm                     {}
    ;

StringData
    : Type2StringOpcode             {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
    | String                        {}
    ;

NamedObject
    : BankFieldTerm                 {}
    | CreateBitFieldTerm            {}
    | CreateByteFieldTerm           {}
    | CreateDWordFieldTerm          {}
    | CreateFieldTerm               {}
```

```
919     | CreateQWordFieldTerm          {}
920     | CreateWordFieldTerm          {}
921     | DataRegionTerm               {}
922     | DeviceTerm                   {}
923     | EventTerm                    {}
924     | FieldTerm                    {}
925     | FunctionTerm                 {}
926     | IndexFieldTerm               {}
927     | MethodTerm                   {}
928     | MutexTerm                    {}
929     | OpRegionTerm                 {}
930     | PowerResTerm                 {}
931     | ProcessorTerm                {}
932     | ThermalZoneTerm              {}
933     ;

935 NameSpaceModifier
936     : AliasTerm                    {}
937     | NameTerm                     {}
938     | ScopeTerm                    {}
939     ;

941 UserTerm
942     : NameString '('               {TrUpdateNode (PARSEOP_METHODCALL, $1);}
943       ArgList ')'                  {$$ = TrLinkChildNode ($1,$4);}
944     ;

946 ArgList
947     :                              {$$ = NULL;}
948     | TermArg
949     | ArgList ','                  /* Allows a trailing comma at list end */
950     | ArgList ','
951       TermArg                      {$$ = TrLinkPeerNode ($1,$3);}
952     ;

954 /*
955 Removed from TermArg due to reduce/reduce conflicts
956     | Type2IntegerOpcode           {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
957     | Type2StringOpcode            {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
958     | Type2BufferOpcode            {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
959     | Type2BufferOrStringOpcode    {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);

961 */

963 TermArg
964     : Type2Opcode                  {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
965     | DataObject                   {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
966     | NameString                   {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
967     | ArgTerm                      {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
968     | LocalTerm                    {$$ = TrSetNodeFlags ($1, NODE_IS_TERM_ARG);
969     ;

971 Target
972     :                              {$$ = TrSetNodeFlags (TrCreateLeafNode (PARS
973     | ','                          {$$ = TrSetNodeFlags (TrCreateLeafNode (PARS
974     | ',' SuperName                {$$ = TrSetNodeFlags ($2, NODE_IS_TARGET);}
975     ;

977 RequiredTarget
978     : ',' SuperName                {$$ = TrSetNodeFlags ($2, NODE_IS_TARGET);}
979     ;

981 SimpleTarget
982     : NameString                   {}
983     | LocalTerm                    {}
984     | ArgTerm                      {}
```



```
985     ;

987 /* Rules for specifying the type of one method argument or return value */

989 ParameterTypePackage
990     :                              {$$ = NULL;}
991     | ObjectTypeKeyword            {$$ = $1;}
992     | ParameterTypePackage ','
993       ObjectTypeKeyword            {$$ = TrLinkPeerNodes (2,$1,$3);}
994     ;

996 ParameterTypePackageList
997     :                              {$$ = NULL;}
998     | ObjectTypeKeyword            {$$ = $1;}
999     | '{' ParameterTypePackage '}' {$$ = $2;}
1000    ;

1002 OptionalParameterTypePackage
1003    :                              {$$ = TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
1004    | ',' ParameterTypePackageList {$$ = TrLinkChildren (TrCreateLeafNode (PARS
1005    ;

1007 /* Rules for specifying the types for method arguments */

1009 ParameterTypesPackage
1010    : ParameterTypePackageList     {$$ = $1;}
1011    | ParameterTypesPackage ','
1012      ParameterTypePackageList     {$$ = TrLinkPeerNodes (2,$1,$3);}
1013    ;

1015 ParameterTypesPackageList
1016    :                              {$$ = NULL;}
1017    | ObjectTypeKeyword            {$$ = $1;}
1018    | '{' ParameterTypesPackage '}' {$$ = $2;}
1019    ;

1021 OptionalParameterTypesPackage
1022    :                              {$$ = TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
1023    | ',' ParameterTypesPackageList {$$ = TrLinkChildren (TrCreateLeafNode (PARS
1024    ;


1027 /* Opcode types */

1029 Type1Opcode
1030    : BreakTerm                    {}
1031    | BreakPointTerm               {}
1032    | ContinueTerm                 {}
1033    | FatalTerm                    {}
1034    | IfElseTerm                   {}
1035    | LoadTerm                     {}
1036    | NoOpTerm                     {}
1037    | NotifyTerm                   {}
1038    | ReleaseTerm                  {}
1039    | ResetTerm                    {}
1040    | ReturnTerm                   {}
1041    | SignalTerm                   {}
1042    | SleepTerm                    {}
1043    | StallTerm                    {}
1044    | SwitchTerm                   {}
1045    | UnloadTerm                   {}
1046    | WhileTerm                    {}
1047    ;

1049 Type2Opcode
1050    : AcquireTerm                  {}
```

```
1051     | CondRefOfTerm                 {}
1052     | CopyObjectTerm                {}
1053     | DerefOfTerm                   {}
1054     | ObjectTypeTerm                {}
1055     | RefOfTerm                     {}
1056     | SizeOfTerm                    {}
1057     | StoreTerm                     {}
1058     | TimerTerm                     {}
1059     | WaitTerm                      {}
1060     | UserTerm                      {}
1061     ;

1063 /*
1064  * Type 3/4/5 opcodes
1065  */

1067 Type2IntegerOpcode                  /* "Type3" opcodes */
1068     : AddTerm                       {}
1069     | AndTerm                       {}
1070     | DecTerm                       {}
1071     | DivideTerm                    {}
1072     | FindSetLeftBitTerm            {}
1073     | FindSetRightBitTerm           {}
1074     | FromBCDTerm                   {}
1075     | IncTerm                       {}
1076     | IndexTerm                     {}
1077     | LAndTerm                      {}
1078     | LEqualTerm                    {}
1079     | LGreaterTerm                  {}
1080     | LGreaterEqualTerm             {}
1081     | LLessTerm                     {}
1082     | LLessEqualTerm                {}
1083     | LNotTerm                      {}
1084     | LNotEqualTerm                 {}
1085     | LoadTableTerm                 {}
1086     | LOrTerm                       {}
1087     | MatchTerm                     {}
1088     | ModTerm                       {}
1089     | MultiplyTerm                  {}
1090     | NAndTerm                      {}
1091     | NOrTerm                       {}
1092     | NotTerm                       {}
1093     | OrTerm                        {}
1094     | ShiftLeftTerm                 {}
1095     | ShiftRightTerm                {}
1096     | SubtractTerm                  {}
1097     | ToBCDTerm                     {}
1098     | ToIntegerTerm                 {}
1099     | XOrTerm                       {}
1100     ;

1102 Type2StringOpcode                  /* "Type4" Opcodes */
1103     : ToDecimalStringTerm           {}
1104     | ToHexStringTerm               {}
1105     | ToStringTerm                  {}
1106     ;

1108 Type2BufferOpcode                  /* "Type5" Opcodes */
1109     : ToBufferTerm                  {}
1110     | ConcatResTerm                 {}
1111     ;

1113 Type2BufferOrStringOpcode
1114     : ConcatTerm                    {}
1115     | MidTerm                       {}
1116     ;
```



```
1118 /*
1119  * A type 3 opcode evaluates to an Integer and cannot have a destination operand
1120  */

1122 Type3Opcode
1123     : EISAIDTerm                    {}
1124     ;

1126 /* Obsolete
1127 Type4Opcode
1128     : ConcatTerm                    {}
1129     | ToDecimalStringTerm           {}
1130     | ToHexStringTerm               {}
1131     | MidTerm                       {}
1132     | ToStringTerm                  {}
1133     ;
1134 */


1137 Type5Opcode
1138     : ResourceTemplateTerm          {}
1139     | UnicodeTerm                   {}
1140     | ToUUIDTerm                    {}
1141     ;

1143 Type6Opcode
1144     : RefOfTerm                     {}
1145     | DerefOfTerm                   {}
1146     | IndexTerm                     {}
1147     | UserTerm                      {}
1148     ;

1150 IncludeTerm
1151     : PARSEOP_INCLUDE '('           {$<n>$ = TrCreateLeafNode (PARSEOP_INCLUDE);
1152         String  ')'                 {TrLinkChildren ($<n>3,1,$4);FlOpenIncludeFi
1153         TermList
1154         IncludeEndTerm              {$$ = TrLinkPeerNodes (3,$<n>3,$7,$8);}
1155     ;

1157 IncludeEndTerm
1158     : PARSEOP_INCLUDE_END           {$$ = TrCreateLeafNode (PARSEOP_INCLUDE_END)
1159     ;

1161 ExternalTerm
1162     : PARSEOP_EXTERNAL '('
1163         NameString
1164         OptionalObjectTypeKeyword
1165         OptionalParameterTypePackage
1166         OptionalParameterTypesPackage
1167         ')'                         {$$ = TrCreateNode (PARSEOP_EXTERNAL,4,$3,$4
1168     | PARSEOP_EXTERNAL '('
1169         error ')'                   {$$ = AslDoError(); yyclearin;}
1170     ;


1173 /******* Named Objects ************************************************/


1176 BankFieldTerm
1177     : PARSEOP_BANKFIELD '('         {$<n>$ = TrCreateLeafNode (PARSEOP_BANKFIELD
1178         NameString
1179         NameStringItem
1180         TermArgItem
1181         ',' AccessTypeKeyword
1182         ',' LockRuleKeyword
```

```
1183         ',' UpdateRuleKeyword
1184         ')' '{'
1185             FieldUnitList '}'       {$$ = TrLinkChildren ($<n>3,7,$4,$5,$6,$8,$1
1186     | PARSEOP_BANKFIELD '('
1187         error ')' '{' error '}'     {$$ = AslDoError(); yyclearin;}
1188     ;

1190 FieldUnitList
1191     :                               {$$ = NULL;}
1192     | FieldUnit
1193     | FieldUnitList ','             /* Allows a trailing comma at list end */
1194     | FieldUnitList ','
1195         FieldUnit                   {$$ = TrLinkPeerNode ($1,$3);}
1196     ;

1198 FieldUnit
1199     : FieldUnitEntry                {}
1200     | OffsetTerm                    {}
1201     | AccessAsTerm                  {}
1202     | ConnectionTerm                {}
1203     ;

1205 FieldUnitEntry
1206     : ',' AmlPackageLengthTerm      {$$ = TrCreateNode (PARSEOP_RESERVED_BYTES,1
1207     | NameSeg ','
1208         AmlPackageLengthTerm        {$$ = TrLinkChildNode ($1,$3);}
1209     ;

1211 OffsetTerm
1212     : PARSEOP_OFFSET '('
1213         AmlPackageLengthTerm
1214         ')'                         {$$ = TrCreateNode (PARSEOP_OFFSET,1,$3);}
1215     | PARSEOP_OFFSET '('
1216         error ')'                   {$$ = AslDoError(); yyclearin;}
1217     ;

1219 AccessAsTerm
1220     : PARSEOP_ACCESSAS '('
1221         AccessTypeKeyword
1222         OptionalAccessAttribTerm
1223         ')'                         {$$ = TrCreateNode (PARSEOP_ACCESSAS,2,$3,$4
1224     | PARSEOP_ACCESSAS '('
1225         error ')'                   {$$ = AslDoError(); yyclearin;}
1226     ;

1228 ConnectionTerm
1229     : PARSEOP_CONNECTION '('
1230         NameString
1231         ')'                         {$$ = TrCreateNode (PARSEOP_CONNECTION,1,$3)
1232     | PARSEOP_CONNECTION '('        {$<n>$ = TrCreateLeafNode (PARSEOP_CONNECTIO
1233         ResourceMacroTerm
1234         ')'                         {$$ = TrLinkChildren ($<n>3, 1,
1235                                             TrLinkChildren (TrCreateLeafNode (PA
1236                                                 TrCreateLeafNode (PARSEOP_DEFAUL
1237                                                 TrCreateLeafNode (PARSEOP_DEFAUL
1238                                                 $4));}
1239     | PARSEOP_CONNECTION '('
1240         error ')'                   {$$ = AslDoError(); yyclearin;}
1241     ;

1243 CreateBitFieldTerm
1244     : PARSEOP_CREATEBITFIELD '('    {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEBIT
1245         TermArg
1246         TermArgItem
1247         NameStringItem
1248         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
```



```
1249     | PARSEOP_CREATEBITFIELD '('
1250         error ')'                   {$$ = AslDoError(); yyclearin;}
1251     ;

1253 CreateByteFieldTerm
1254     : PARSEOP_CREATEBYTEFIELD '('   {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEBYT
1255         TermArg
1256         TermArgItem
1257         NameStringItem
1258         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
1259     | PARSEOP_CREATEBYTEFIELD '('
1260         error ')'                   {$$ = AslDoError(); yyclearin;}
1261     ;

1263 CreateDWordFieldTerm
1264     : PARSEOP_CREATEDWORDFIELD '('  {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEDWO
1265         TermArg
1266         TermArgItem
1267         NameStringItem
1268         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
1269     | PARSEOP_CREATEDWORDFIELD '('
1270         error ')'                   {$$ = AslDoError(); yyclearin;}
1271     ;

1273 CreateFieldTerm
1274     : PARSEOP_CREATEFIELD '('       {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEFIE
1275         TermArg
1276         TermArgItem
1277         TermArgItem
1278         NameStringItem
1279         ')'                         {$$ = TrLinkChildren ($<n>3,4,$4,$5,$6,TrSet
1280     | PARSEOP_CREATEFIELD '('
1281         error ')'                   {$$ = AslDoError(); yyclearin;}
1282     ;

1284 CreateQWordFieldTerm
1285     : PARSEOP_CREATEQWORDFIELD '('  {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEQWO
1286         TermArg
1287         TermArgItem
1288         NameStringItem
1289         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
1290     | PARSEOP_CREATEQWORDFIELD '('
1291         error ')'                   {$$ = AslDoError(); yyclearin;}
1292     ;

1294 CreateWordFieldTerm
1295     : PARSEOP_CREATEWORDFIELD '('   {$<n>$ = TrCreateLeafNode (PARSEOP_CREATEWOR
1296         TermArg
1297         TermArgItem
1298         NameStringItem
1299         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
1300     | PARSEOP_CREATEWORDFIELD '('
1301         error ')'                   {$$ = AslDoError(); yyclearin;}
1302     ;

1304 DataRegionTerm
1305     : PARSEOP_DATATABLEREGION '('   {$<n>$ = TrCreateLeafNode (PARSEOP_DATATABLE
1306         NameString
1307         TermArgItem
1308         TermArgItem
1309         TermArgItem
1310         ')'                         {$$ = TrLinkChildren ($<n>3,4,TrSetNodeFlags
1311     | PARSEOP_DATATABLEREGION '('
1312         error ')'                   {$$ = AslDoError(); yyclearin;}
1313     ;
```

```
1315 DeviceTerm
1316     : PARSEOP_DEVICE '('            {$<n>$ = TrCreateLeafNode (PARSEOP_DEVICE);}
1317         NameString
1318         ')' '{'
1319             ObjectList '}'          {$$ = TrLinkChildren ($<n>3,2,TrSetNodeFlags
1320     | PARSEOP_DEVICE '('
1321         error ')'                   {$$ = AslDoError(); yyclearin;}
1322     ;

1324 EventTerm
1325     : PARSEOP_EVENT '('             {$<n>$ = TrCreateLeafNode (PARSEOP_EVENT);}
1326         NameString
1327         ')'                         {$$ = TrLinkChildren ($<n>3,1,TrSetNodeFlags
1328     | PARSEOP_EVENT '('
1329         error ')'                   {$$ = AslDoError(); yyclearin;}
1330     ;

1332 FieldTerm
1333     : PARSEOP_FIELD '('             {$<n>$ = TrCreateLeafNode (PARSEOP_FIELD);}
1334         NameString
1335         ',' AccessTypeKeyword
1336         ',' LockRuleKeyword
1337         ',' UpdateRuleKeyword
1338         ')' '{'
1339             FieldUnitList '}'       {$$ = TrLinkChildren ($<n>3,5,$4,$6,$8,$10,$
1340     | PARSEOP_FIELD '('
1341         error ')' '{' error '}'     {$$ = AslDoError(); yyclearin;}
1342     ;

1344 FunctionTerm
1345     : PARSEOP_FUNCTION '('          {$<n>$ = TrCreateLeafNode (PARSEOP_METHOD);}
1346         NameString
1347         OptionalParameterTypePackage
1348         OptionalParameterTypesPackage
1349         ')' '{'
1350             TermList '}'            {$$ = TrLinkChildren ($<n>3,7,TrSetNodeFlags
1351                                         TrCreateValuedLeafNode (PARSEOP_BYTECONS
1352                                         TrCreateLeafNode (PARSEOP_SERIALIZERULE_
1353                                         TrCreateValuedLeafNode (PARSEOP_BYTECONS
1354     | PARSEOP_FUNCTION '('
1355         error ')'                   {$$ = AslDoError(); yyclearin;}
1356     ;

1358 IndexFieldTerm
1359     : PARSEOP_INDEXFIELD '('        {$<n>$ = TrCreateLeafNode (PARSEOP_INDEXFIEL
1360         NameString
1361         NameStringItem
1362         ',' AccessTypeKeyword
1363         ',' LockRuleKeyword
1364         ',' UpdateRuleKeyword
1365         ')' '{'
1366             FieldUnitList '}'       {$$ = TrLinkChildren ($<n>3,6,$4,$5,$7,$9,$1
1367     | PARSEOP_INDEXFIELD '('
1368         error ')' '{' error '}'     {$$ = AslDoError(); yyclearin;}
1369     ;

1371 MethodTerm
1372     : PARSEOP_METHOD  '('           {$<n>$ = TrCreateLeafNode (PARSEOP_METHOD);}
1373         NameString
1374         OptionalByteConstExpr       {UtCheckIntegerRange ($5, 0, 7);}
1375         OptionalSerializeRuleKeyword
1376         OptionalByteConstExpr
1377         OptionalParameterTypePackage
1378         OptionalParameterTypesPackage
1379         ')' '{'
1380             TermList '}'            {$$ = TrLinkChildren ($<n>3,7,TrSetNodeFlags
```



```
1381     | PARSEOP_METHOD '('
1382         error ')'                   {$$ = AslDoError(); yyclearin;}
1383     ;

1385 MutexTerm
1386     : PARSEOP_MUTEX '('             {$<n>$ = TrCreateLeafNode (PARSEOP_MUTEX);}
1387         NameString
1388         ',' ByteConstExpr
1389         ')'                         {$$ = TrLinkChildren ($<n>3,2,TrSetNodeFlags
1390     | PARSEOP_MUTEX '('
1391         error ')'                   {$$ = AslDoError(); yyclearin;}
1392     ;

1394 OpRegionTerm
1395     : PARSEOP_OPERATIONREGION '('   {$<n>$ = TrCreateLeafNode (PARSEOP_OPERATION
1396         NameString
1397         ',' OpRegionSpaceIdTerm
1398         TermArgItem
1399         TermArgItem
1400         ')'                         {$$ = TrLinkChildren ($<n>3,4,TrSetNodeFlags
1401     | PARSEOP_OPERATIONREGION '('
1402         error ')'                   {$$ = AslDoError(); yyclearin;}
1403     ;

1405 OpRegionSpaceIdTerm
1406     : RegionSpaceKeyword            {}
1407     | ByteConst                     {$$ = UtCheckIntegerRange ($1, 0x80, 0xFF);}
1408     ;

1410 PowerResTerm
1411     : PARSEOP_POWERRESOURCE '('     {$<n>$ = TrCreateLeafNode (PARSEOP_POWERRESO
1412         NameString
1413         ',' ByteConstExpr
1414         ',' WordConstExpr
1415         ')' '{'
1416             ObjectList '}'          {$$ = TrLinkChildren ($<n>3,4,TrSetNodeFlags
1417     | PARSEOP_POWERRESOURCE '('
1418         error ')'                   {$$ = AslDoError(); yyclearin;}
1419     ;

1421 ProcessorTerm
1422     : PARSEOP_PROCESSOR '('         {$<n>$ = TrCreateLeafNode (PARSEOP_PROCESSOR
1423         NameString
1424         ',' ByteConstExpr
1425         OptionalDWordConstExpr
1426         OptionalByteConstExpr
1427         ')' '{'
1428             ObjectList '}'          {$$ = TrLinkChildren ($<n>3,5,TrSetNodeFlags
1429     | PARSEOP_PROCESSOR '('
1430         error ')'                   {$$ = AslDoError(); yyclearin;}
1431     ;

1433 ThermalZoneTerm
1434     : PARSEOP_THERMALZONE '('       {$<n>$ = TrCreateLeafNode (PARSEOP_THERMALZO
1435         NameString
1436         ')' '{'
1437             ObjectList '}'          {$$ = TrLinkChildren ($<n>3,2,TrSetNodeFlags
1438     | PARSEOP_THERMALZONE '('
1439         error ')'                   {$$ = AslDoError(); yyclearin;}
1440     ;


1443 /******* Namespace modifiers *************************************************/


1446 AliasTerm
```

```
1447     : PARSEOP_ALIAS '('            {$<n>$ = TrCreateLeafNode (PARSEOP_ALIAS);}
1448         NameString
1449         NameStringItem
1450         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,TrSetNodeFl
1451     | PARSEOP_ALIAS '('
1452         error ')'                  {$$ = AslDoError(); yyclearin;}
1453     ;

1455 NameTerm
1456     : PARSEOP_NAME '('             {$<n>$ = TrCreateLeafNode (PARSEOP_NAME);}
1457         NameString
1458         ',' DataObject
1459         ')'                        {$$ = TrLinkChildren ($<n>3,2,TrSetNodeFlags
1460     | PARSEOP_NAME '('
1461         error ')'                  {$$ = AslDoError(); yyclearin;}
1462     ;

1464 ScopeTerm
1465     : PARSEOP_SCOPE '('            {$<n>$ = TrCreateLeafNode (PARSEOP_SCOPE);}
1466         NameString
1467         ')' '{'
1468             ObjectList '}'         {$$ = TrLinkChildren ($<n>3,2,TrSetNodeFlags
1469     | PARSEOP_SCOPE '('
1470         error ')'                  {$$ = AslDoError(); yyclearin;}
1471     ;


1474 /******* Type 1 opcodes ************************************************/


1477 BreakTerm
1478     : PARSEOP_BREAK                {$$ = TrCreateNode (PARSEOP_BREAK, 0);}
1479     ;

1481 BreakPointTerm
1482     : PARSEOP_BREAKPOINT           {$$ = TrCreateNode (PARSEOP_BREAKPOINT, 0);}
1483     ;

1485 ContinueTerm
1486     : PARSEOP_CONTINUE             {$$ = TrCreateNode (PARSEOP_CONTINUE, 0);}
1487     ;

1489 FatalTerm
1490     : PARSEOP_FATAL '('            {$<n>$ = TrCreateLeafNode (PARSEOP_FATAL);}
1491         ByteConstExpr
1492         ',' DWordConstExpr
1493         TermArgItem
1494         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$6,$7);}
1495     | PARSEOP_FATAL '('
1496         error ')'                  {$$ = AslDoError(); yyclearin;}
1497     ;

1499 IfElseTerm
1500     : IfTerm ElseTerm              {$$ = TrLinkPeerNode ($1,$2);}
1501     ;

1503 IfTerm
1504     : PARSEOP_IF '('               {$<n>$ = TrCreateLeafNode (PARSEOP_IF);}
1505         TermArg
1506         ')' '{'
1507             TermList '}'           {$$ = TrLinkChildren ($<n>3,2,$4,$7);}

1509     | PARSEOP_IF '('
1510         error ')'                  {$$ = AslDoError(); yyclearin;}
1511     ;
```



```
1513 ElseTerm
1514     :                              {$$ = NULL;}
1515     | PARSEOP_ELSE '{'             {$<n>$ = TrCreateLeafNode (PARSEOP_ELSE);}
1516         TermList '}'               {$$ = TrLinkChildren ($<n>3,1,$4);}

1518     | PARSEOP_ELSE '{'
1519         error '}'                  {$$ = AslDoError(); yyclearin;}

1521     | PARSEOP_ELSE
1522         error                      {$$ = AslDoError(); yyclearin;}

1524     | PARSEOP_ELSEIF '('           {$<n>$ = TrCreateLeafNode (PARSEOP_ELSE);}
1525         TermArg                    {$<n>$ = TrCreateLeafNode (PARSEOP_IF);}
1526         ')' '{'
1527             TermList '}'           {TrLinkChildren ($<n>5,2,$4,$8);}
1528         ElseTerm                   {TrLinkPeerNode ($<n>5,$11);}
1529                                    {$$ = TrLinkChildren ($<n>3,1,$<n>5);}

1531     | PARSEOP_ELSEIF '('
1532         error ')'                  {$$ = AslDoError(); yyclearin;}

1534     | PARSEOP_ELSEIF
1535         error                      {$$ = AslDoError(); yyclearin;}
1536     ;

1538 LoadTerm
1539     : PARSEOP_LOAD '('             {$<n>$ = TrCreateLeafNode (PARSEOP_LOAD);}
1540         NameString
1541         RequiredTarget
1542         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1543     | PARSEOP_LOAD '('
1544         error ')'                  {$$ = AslDoError(); yyclearin;}
1545     ;

1547 NoOpTerm
1548     : PARSEOP_NOOP                 {$$ = TrCreateNode (PARSEOP_NOOP, 0);}
1549     ;

1551 NotifyTerm
1552     : PARSEOP_NOTIFY '('           {$<n>$ = TrCreateLeafNode (PARSEOP_NOTIFY);}
1553         SuperName
1554         TermArgItem
1555         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1556     | PARSEOP_NOTIFY '('
1557         error ')'                  {$$ = AslDoError(); yyclearin;}
1558     ;

1560 ReleaseTerm
1561     : PARSEOP_RELEASE '('          {$<n>$ = TrCreateLeafNode (PARSEOP_RELEASE);
1562         SuperName
1563         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1564     | PARSEOP_RELEASE '('
1565         error ')'                  {$$ = AslDoError(); yyclearin;}
1566     ;

1568 ResetTerm
1569     : PARSEOP_RESET '('            {$<n>$ = TrCreateLeafNode (PARSEOP_RESET);}
1570         SuperName
1571         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1572     | PARSEOP_RESET '('
1573         error ')'                  {$$ = AslDoError(); yyclearin;}
1574     ;

1576 ReturnTerm
1577     : PARSEOP_RETURN '('           {$<n>$ = TrCreateLeafNode (PARSEOP_RETURN);}
1578         OptionalReturnArg
```

```
1579         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1580     | PARSEOP_RETURN               {$$ = TrLinkChildren (TrCreateLeafNode (PARS
1581     | PARSEOP_RETURN '('
1582         error ')'                  {$$ = AslDoError(); yyclearin;}
1583     ;

1585 SignalTerm
1586     : PARSEOP_SIGNAL '('           {$<n>$ = TrCreateLeafNode (PARSEOP_SIGNAL);}
1587         SuperName
1588         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1589     | PARSEOP_SIGNAL '('
1590         error ')'                  {$$ = AslDoError(); yyclearin;}
1591     ;

1593 SleepTerm
1594     : PARSEOP_SLEEP '('            {$<n>$ = TrCreateLeafNode (PARSEOP_SLEEP);}
1595         TermArg
1596         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1597     | PARSEOP_SLEEP '('
1598         error ')'                  {$$ = AslDoError(); yyclearin;}
1599     ;

1601 StallTerm
1602     : PARSEOP_STALL '('            {$<n>$ = TrCreateLeafNode (PARSEOP_STALL);}
1603         TermArg
1604         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1605     | PARSEOP_STALL '('
1606         error ')'                  {$$ = AslDoError(); yyclearin;}
1607     ;

1609 SwitchTerm
1610     : PARSEOP_SWITCH '('           {$<n>$ = TrCreateLeafNode (PARSEOP_SWITCH);}
1611         TermArg
1612         ')' '{'
1613             CaseDefaultTermList '}'
1614                                    {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
1615     | PARSEOP_SWITCH '('
1616         error ')'                  {$$ = AslDoError(); yyclearin;}
1617     ;

1619 /*
1620  * Case-Default list; allow only one Default term and unlimited Case terms
1621  */

1623 CaseDefaultTermList
1624     :                              {$$ = NULL;}
1625     | CaseTerm  {}
1626     | DefaultTerm   {}
1627     | CaseDefaultTermList
1628         CaseTerm                   {$$ = TrLinkPeerNode ($1,$2);}
1629     | CaseDefaultTermList
1630         DefaultTerm                {$$ = TrLinkPeerNode ($1,$2);}

1632 /* Original - attempts to force zero or one default term within the switch */

1634 /*
1635 CaseDefaultTermList
1636     :                              {$$ = NULL;}
1637     | CaseTermList
1638         DefaultTerm
1639         CaseTermList               {$$ = TrLinkPeerNode ($1,TrLinkPeerNode ($2,
1640     | CaseTermList
1641         CaseTerm                   {$$ = TrLinkPeerNode ($1,$2);}
1642     ;

1644 CaseTermList
```



```
1645     :                              {$$ = NULL;}
1646     | CaseTerm                     {}
1647     | CaseTermList
1648         CaseTerm                   {$$ = TrLinkPeerNode ($1,$2);}
1649     ;
1650 */

1652 CaseTerm
1653     : PARSEOP_CASE '('             {$<n>$ = TrCreateLeafNode (PARSEOP_CASE);}
1654         DataObject
1655         ')' '{'
1656             TermList '}'           {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
1657     | PARSEOP_CASE '('
1658         error ')'                  {$$ = AslDoError(); yyclearin;}
1659     ;

1661 DefaultTerm
1662     : PARSEOP_DEFAULT '{'          {$<n>$ = TrCreateLeafNode (PARSEOP_DEFAULT);
1663         TermList '}'               {$$ = TrLinkChildren ($<n>3,1,$4);}
1664     | PARSEOP_DEFAULT '{'
1665         error '}'                  {$$ = AslDoError(); yyclearin;}
1666     ;

1668 UnloadTerm
1669     : PARSEOP_UNLOAD '('           {$<n>$ = TrCreateLeafNode (PARSEOP_UNLOAD);}
1670         SuperName
1671         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1672     | PARSEOP_UNLOAD '('
1673         error ')'                  {$$ = AslDoError(); yyclearin;}
1674     ;

1676 WhileTerm
1677     : PARSEOP_WHILE '('            {$<n>$ = TrCreateLeafNode (PARSEOP_WHILE);}
1678         TermArg
1679         ')' '{' TermList '}'
1680                                    {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
1681     | PARSEOP_WHILE '('
1682         error ')'                  {$$ = AslDoError(); yyclearin;}
1683     ;


1686 /******* Type 2 opcodes ************************************************/

1688 AcquireTerm
1689     : PARSEOP_ACQUIRE '('          {$<n>$ = TrCreateLeafNode (PARSEOP_ACQUIRE);
1690         SuperName
1691         ',' WordConstExpr
1692         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$6);}
1693     | PARSEOP_ACQUIRE '('
1694         error ')'                  {$$ = AslDoError(); yyclearin;}
1695     ;

1697 AddTerm
1698     : PARSEOP_ADD '('              {$<n>$ = TrCreateLeafNode (PARSEOP_ADD);}
1699         TermArg
1700         TermArgItem
1701         Target
1702         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1703     | PARSEOP_ADD '('
1704         error ')'                  {$$ = AslDoError(); yyclearin;}
1705     ;

1707 AndTerm
1708     : PARSEOP_AND '('              {$<n>$ = TrCreateLeafNode (PARSEOP_AND);}
1709         TermArg
1710         TermArgItem
```

```
1711         Target
1712         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1713     | PARSEOP_AND '('
1714         error ')'                  {$$ = AslDoError(); yyclearin;}
1715     ;

1717 ConcatTerm
1718     : PARSEOP_CONCATENATE '('       {$<n>$ = TrCreateLeafNode (PARSEOP_CONCATENA
1719         TermArg
1720         TermArgItem
1721         Target
1722         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1723     | PARSEOP_CONCATENATE '('
1724         error ')'                  {$$ = AslDoError(); yyclearin;}
1725     ;

1727 ConcatResTerm
1728     : PARSEOP_CONCATENATERESTEMPLATE '('    {$<n>$ = TrCreateLeafNode (PARSEOP_C
1729         TermArg
1730         TermArgItem
1731         Target
1732         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1733     | PARSEOP_CONCATENATERESTEMPLATE '('
1734         error ')'                  {$$ = AslDoError(); yyclearin;}
1735     ;

1737 CondRefOfTerm
1738     : PARSEOP_CONDREFOF '('         {$<n>$ = TrCreateLeafNode (PARSEOP_CONDREFOF
1739         SuperName
1740         Target
1741         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1742     | PARSEOP_CONDREFOF '('
1743         error ')'                  {$$ = AslDoError(); yyclearin;}
1744     ;

1746 CopyObjectTerm
1747     : PARSEOP_COPYOBJECT '('        {$<n>$ = TrCreateLeafNode (PARSEOP_COPYOBJEC
1748         TermArg
1749         ',' SimpleTarget
1750         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,TrSetNodeFl
1751     | PARSEOP_COPYOBJECT '('
1752         error ')'                  {$$ = AslDoError(); yyclearin;}
1753     ;

1755 DecTerm
1756     : PARSEOP_DECREMENT '('         {$<n>$ = TrCreateLeafNode (PARSEOP_DECREMENT
1757         SuperName
1758         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1759     | PARSEOP_DECREMENT '('
1760         error ')'                  {$$ = AslDoError(); yyclearin;}
1761     ;

1763 DerefOfTerm
1764     : PARSEOP_DEREFOF '('           {$<n>$ = TrCreateLeafNode (PARSEOP_DEREFOF);
1765         TermArg
1766         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1767     | PARSEOP_DEREFOF '('
1768         error ')'                  {$$ = AslDoError(); yyclearin;}
1769     ;

1771 DivideTerm
1772     : PARSEOP_DIVIDE '('            {$<n>$ = TrCreateLeafNode (PARSEOP_DIVIDE);}
1773         TermArg
1774         TermArgItem
1775         Target
1776         Target
```



```
1777         ')'                        {$$ = TrLinkChildren ($<n>3,4,$4,$5,$6,$7);}
1778     | PARSEOP_DIVIDE '('
1779         error ')'                  {$$ = AslDoError(); yyclearin;}
1780     ;

1782 FindSetLeftBitTerm
1783     : PARSEOP_FINDSETLEFTBIT '('    {$<n>$ = TrCreateLeafNode (PARSEOP_FINDSETLE
1784         TermArg
1785         Target
1786         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1787     | PARSEOP_FINDSETLEFTBIT '('
1788         error ')'                  {$$ = AslDoError(); yyclearin;}
1789     ;

1791 FindSetRightBitTerm
1792     : PARSEOP_FINDSETRIGHTBIT '('   {$<n>$ = TrCreateLeafNode (PARSEOP_FINDSETRI
1793         TermArg
1794         Target
1795         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1796     | PARSEOP_FINDSETRIGHTBIT '('
1797         error ')'                  {$$ = AslDoError(); yyclearin;}
1798     ;

1800 FromBCDTerm
1801     : PARSEOP_FROMBCD '('           {$<n>$ = TrCreateLeafNode (PARSEOP_FROMBCD);
1802         TermArg
1803         Target
1804         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1805     | PARSEOP_FROMBCD '('
1806         error ')'                  {$$ = AslDoError(); yyclearin;}
1807     ;

1809 IncTerm
1810     : PARSEOP_INCREMENT '('         {$<n>$ = TrCreateLeafNode (PARSEOP_INCREMENT
1811         SuperName
1812         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
1813     | PARSEOP_INCREMENT '('
1814         error ')'                  {$$ = AslDoError(); yyclearin;}
1815     ;

1817 IndexTerm
1818     : PARSEOP_INDEX '('             {$<n>$ = TrCreateLeafNode (PARSEOP_INDEX);}
1819         TermArg
1820         TermArgItem
1821         Target
1822         ')'                        {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1823     | PARSEOP_INDEX '('
1824         error ')'                  {$$ = AslDoError(); yyclearin;}
1825     ;

1827 LAndTerm
1828     : PARSEOP_LAND '('              {$<n>$ = TrCreateLeafNode (PARSEOP_LAND);}
1829         TermArg
1830         TermArgItem
1831         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1832     | PARSEOP_LAND '('
1833         error ')'                  {$$ = AslDoError(); yyclearin;}
1834     ;

1836 LEqualTerm
1837     : PARSEOP_LEQUAL '('            {$<n>$ = TrCreateLeafNode (PARSEOP_LEQUAL);}
1838         TermArg
1839         TermArgItem
1840         ')'                        {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1841     | PARSEOP_LEQUAL '('
1842         error ')'                  {$$ = AslDoError(); yyclearin;}
```

```
1843     ;

1845 LGreaterTerm
1846     : PARSEOP_LGREATER '('          {$<n>$ = TrCreateLeafNode (PARSEOP_LGREATER)
1847         TermArg
1848         TermArgItem
1849         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1850     | PARSEOP_LGREATER '('
1851         error ')'                   {$$ = AslDoError(); yyclearin;}
1852     ;

1854 LGreaterEqualTerm
1855     : PARSEOP_LGREATEREQUAL '('     {$<n>$ = TrCreateLeafNode (PARSEOP_LLESS);}
1856         TermArg
1857         TermArgItem
1858         ')'                         {$$ = TrCreateNode (PARSEOP_LNOT, 1, TrLinkC
1859     | PARSEOP_LGREATEREQUAL '('
1860         error ')'                   {$$ = AslDoError(); yyclearin;}
1861     ;

1863 LLessTerm
1864     : PARSEOP_LLESS '('             {$<n>$ = TrCreateLeafNode (PARSEOP_LLESS);}
1865         TermArg
1866         TermArgItem
1867         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1868     | PARSEOP_LLESS '('
1869         error ')'                   {$$ = AslDoError(); yyclearin;}
1870     ;

1872 LLessEqualTerm
1873     : PARSEOP_LLESSEQUAL '('        {$<n>$ = TrCreateLeafNode (PARSEOP_LGREATER)
1874         TermArg
1875         TermArgItem
1876         ')'                         {$$ = TrCreateNode (PARSEOP_LNOT, 1, TrLinkC
1877     | PARSEOP_LLESSEQUAL '('
1878         error ')'                   {$$ = AslDoError(); yyclearin;}
1879     ;

1881 LNotTerm
1882     : PARSEOP_LNOT '('              {$<n>$ = TrCreateLeafNode (PARSEOP_LNOT);}
1883         TermArg
1884         ')'                         {$$ = TrLinkChildren ($<n>3,1,$4);}
1885     | PARSEOP_LNOT '('
1886         error ')'                   {$$ = AslDoError(); yyclearin;}
1887     ;

1889 LNotEqualTerm
1890     : PARSEOP_LNOTEQUAL '('         {$<n>$ = TrCreateLeafNode (PARSEOP_LEQUAL);}
1891         TermArg
1892         TermArgItem
1893         ')'                         {$$ = TrCreateNode (PARSEOP_LNOT, 1, TrLinkC
1894     | PARSEOP_LNOTEQUAL '('
1895         error ')'                   {$$ = AslDoError(); yyclearin;}
1896     ;

1898 LoadTableTerm
1899     : PARSEOP_LOADTABLE '('         {$<n>$ = TrCreateLeafNode (PARSEOP_LOADTABLE
1900         TermArg
1901         TermArgItem
1902         TermArgItem
1903         OptionalListString
1904         OptionalListString
1905         OptionalReference
1906         ')'                         {$$ = TrLinkChildren ($<n>3,6,$4,$5,$6,$7,$8
1907     | PARSEOP_LOADTABLE '('
1908         error ')'                   {$$ = AslDoError(); yyclearin;}
```



```
1909     ;

1911 LOrTerm
1912     : PARSEOP_LOR '('               {$<n>$ = TrCreateLeafNode (PARSEOP_LOR);}
1913         TermArg
1914         TermArgItem
1915         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1916     | PARSEOP_LOR '('
1917         error ')'                   {$$ = AslDoError(); yyclearin;}
1918     ;

1920 MatchTerm
1921     : PARSEOP_MATCH '('             {$<n>$ = TrCreateLeafNode (PARSEOP_MATCH);}
1922         TermArg
1923         ',' MatchOpKeyword
1924         TermArgItem
1925         ',' MatchOpKeyword
1926         TermArgItem
1927         TermArgItem
1928         ')'                         {$$ = TrLinkChildren ($<n>3,6,$4,$6,$7,$9,$1
1929     | PARSEOP_MATCH '('
1930         error ')'                   {$$ = AslDoError(); yyclearin;}
1931     ;

1933 MidTerm
1934     : PARSEOP_MID '('               {$<n>$ = TrCreateLeafNode (PARSEOP_MID);}
1935         TermArg
1936         TermArgItem
1937         TermArgItem
1938         Target
1939         ')'                         {$$ = TrLinkChildren ($<n>3,4,$4,$5,$6,$7);}
1940     | PARSEOP_MID '('
1941         error ')'                   {$$ = AslDoError(); yyclearin;}
1942     ;

1944 ModTerm
1945     : PARSEOP_MOD '('               {$<n>$ = TrCreateLeafNode (PARSEOP_MOD);}
1946         TermArg
1947         TermArgItem
1948         Target
1949         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1950     | PARSEOP_MOD '('
1951         error ')'                   {$$ = AslDoError(); yyclearin;}
1952     ;

1954 MultiplyTerm
1955     : PARSEOP_MULTIPLY '('          {$<n>$ = TrCreateLeafNode (PARSEOP_MULTIPLY)
1956         TermArg
1957         TermArgItem
1958         Target
1959         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1960     | PARSEOP_MULTIPLY '('
1961         error ')'                   {$$ = AslDoError(); yyclearin;}
1962     ;

1964 NAndTerm
1965     : PARSEOP_NAND '('              {$<n>$ = TrCreateLeafNode (PARSEOP_NAND);}
1966         TermArg
1967         TermArgItem
1968         Target
1969         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1970     | PARSEOP_NAND '('
1971         error ')'                   {$$ = AslDoError(); yyclearin;}
1972     ;

1974 NOrTerm
```

```
1975     : PARSEOP_NOR '('               {$<n>$ = TrCreateLeafNode (PARSEOP_NOR);}
1976         TermArg
1977         TermArgItem
1978         Target
1979         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
1980     | PARSEOP_NOR '('
1981         error ')'                   {$$ = AslDoError(); yyclearin;}
1982     ;

1984 NotTerm
1985     : PARSEOP_NOT '('               {$<n>$ = TrCreateLeafNode (PARSEOP_NOT);}
1986         TermArg
1987         Target
1988         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
1989     | PARSEOP_NOT '('
1990         error ')'                   {$$ = AslDoError(); yyclearin;}
1991     ;

1993 ObjectTypeTerm
1994     : PARSEOP_OBJECTTYPE '('        {$<n>$ = TrCreateLeafNode (PARSEOP_OBJECTTYP
1995         ObjectTypeName
1996         ')'                         {$$ = TrLinkChildren ($<n>3,1,$4);}
1997     | PARSEOP_OBJECTTYPE '('
1998         error ')'                   {$$ = AslDoError(); yyclearin;}
1999     ;

2001 OrTerm
2002     : PARSEOP_OR '('                {$<n>$ = TrCreateLeafNode (PARSEOP_OR);}
2003         TermArg
2004         TermArgItem
2005         Target
2006         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2007     | PARSEOP_OR '('
2008         error ')'                   {$$ = AslDoError(); yyclearin;}
2009     ;

2011 /*
2012  * In RefOf, the node isn't really a target, but we can't keep track of it after
2013  * we've taken a pointer to it. (hard to tell if a local becomes initialized thi
2014  */
2015 RefOfTerm
2016     : PARSEOP_REFOF '('             {$<n>$ = TrCreateLeafNode (PARSEOP_REFOF);}
2017         SuperName
2018         ')'                         {$$ = TrLinkChildren ($<n>3,1,TrSetNodeFlags
2019     | PARSEOP_REFOF '('
2020         error ')'                   {$$ = AslDoError(); yyclearin;}
2021     ;

2023 ShiftLeftTerm
2024     : PARSEOP_SHIFTLEFT '('         {$<n>$ = TrCreateLeafNode (PARSEOP_SHIFTLEFT
2025         TermArg
2026         TermArgItem
2027         Target
2028         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2029     | PARSEOP_SHIFTLEFT '('
2030         error ')'                   {$$ = AslDoError(); yyclearin;}
2031     ;

2033 ShiftRightTerm
2034     : PARSEOP_SHIFTRIGHT '('        {$<n>$ = TrCreateLeafNode (PARSEOP_SHIFTRIGH
2035         TermArg
2036         TermArgItem
2037         Target
2038         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2039     | PARSEOP_SHIFTRIGHT '('
2040         error ')'                   {$$ = AslDoError(); yyclearin;}
```



```
2041     ;

2043 SizeOfTerm
2044     : PARSEOP_SIZEOF '('            {$<n>$ = TrCreateLeafNode (PARSEOP_SIZEOF);}
2045         SuperName
2046         ')'                         {$$ = TrLinkChildren ($<n>3,1,$4);}
2047     | PARSEOP_SIZEOF '('
2048         error ')'                   {$$ = AslDoError(); yyclearin;}
2049     ;

2051 StoreTerm
2052     : PARSEOP_STORE '('             {$<n>$ = TrCreateLeafNode (PARSEOP_STORE);}
2053         TermArg
2054         ',' SuperName
2055         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,TrSetNodeFl
2056     | PARSEOP_STORE '('
2057         error ')'                   {$$ = AslDoError(); yyclearin;}
2058     ;

2060 SubtractTerm
2061     : PARSEOP_SUBTRACT '('          {$<n>$ = TrCreateLeafNode (PARSEOP_SUBTRACT)
2062         TermArg
2063         TermArgItem
2064         Target
2065         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2066     | PARSEOP_SUBTRACT '('
2067         error ')'                   {$$ = AslDoError(); yyclearin;}
2068     ;

2070 TimerTerm
2071     : PARSEOP_TIMER '('             {$<n>$ = TrCreateLeafNode (PARSEOP_TIMER);}
2072         ')'                         {$$ = TrLinkChildren ($<n>3,0);}
2073     | PARSEOP_TIMER                 {$$ = TrLinkChildren (TrCreateLeafNode (PARS
2074     | PARSEOP_TIMER '('
2075         error ')'                   {$$ = AslDoError(); yyclearin;}
2076     ;

2078 ToBCDTerm
2079     : PARSEOP_TOBCD '('             {$<n>$ = TrCreateLeafNode (PARSEOP_TOBCD);}
2080         TermArg
2081         Target
2082         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2083     | PARSEOP_TOBCD '('
2084         error ')'                   {$$ = AslDoError(); yyclearin;}
2085     ;

2087 ToBufferTerm
2088     : PARSEOP_TOBUFFER '('          {$<n>$ = TrCreateLeafNode (PARSEOP_TOBUFFER)
2089         TermArg
2090         Target
2091         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2092     | PARSEOP_TOBUFFER '('
2093         error ')'                   {$$ = AslDoError(); yyclearin;}
2094     ;

2096 ToDecimalStringTerm
2097     : PARSEOP_TODECIMALSTRING '('   {$<n>$ = TrCreateLeafNode (PARSEOP_TODECIMAL
2098         TermArg
2099         Target
2100         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2101     | PARSEOP_TODECIMALSTRING '('
2102         error ')'                   {$$ = AslDoError(); yyclearin;}
2103     ;

2105 ToHexStringTerm
2106     : PARSEOP_TOHEXSTRING '('       {$<n>$ = TrCreateLeafNode (PARSEOP_TOHEXSTRI
```

```
2107         TermArg
2108         Target
2109         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2110     | PARSEOP_TOHEXSTRING '('
2111         error ')'                   {$$ = AslDoError(); yyclearin;}
2112     ;

2114 ToIntegerTerm
2115     : PARSEOP_TOINTEGER '('         {$<n>$ = TrCreateLeafNode (PARSEOP_TOINTEGER
2116         TermArg
2117         Target
2118         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2119     | PARSEOP_TOINTEGER '('
2120         error ')'                   {$$ = AslDoError(); yyclearin;}
2121     ;

2123 ToStringTerm
2124     : PARSEOP_TOSTRING '('          {$<n>$ = TrCreateLeafNode (PARSEOP_TOSTRING)
2125         TermArg
2126         OptionalCount
2127         Target
2128         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2129     | PARSEOP_TOSTRING '('
2130         error ')'                   {$$ = AslDoError(); yyclearin;}
2131     ;

2133 ToUUIDTerm
2134     : PARSEOP_TOUUID '('
2135         StringData ')'              {$$ = TrUpdateNode (PARSEOP_TOUUID, $3);}
2136     | PARSEOP_TOUUID '('
2137         error ')'                   {$$ = AslDoError(); yyclearin;}
2138     ;

2140 WaitTerm
2141     : PARSEOP_WAIT '('              {$<n>$ = TrCreateLeafNode (PARSEOP_WAIT);}
2142         SuperName
2143         TermArgItem
2144         ')'                         {$$ = TrLinkChildren ($<n>3,2,$4,$5);}
2145     | PARSEOP_WAIT '('
2146         error ')'                   {$$ = AslDoError(); yyclearin;}
2147     ;

2149 XOrTerm
2150     : PARSEOP_XOR '('               {$<n>$ = TrCreateLeafNode (PARSEOP_XOR);}
2151         TermArg
2152         TermArgItem
2153         Target
2154         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$5,$6);}
2155     | PARSEOP_XOR '('
2156         error ')'                   {$$ = AslDoError(); yyclearin;}
2157     ;


2160 /******* Keywords ****************************************************/


2163 AccessAttribKeyword
2164     : PARSEOP_ACCESSATTRIB_BLOCK            {$$ = TrCreateLeafNode (PARSEOP_ACCE
2165     | PARSEOP_ACCESSATTRIB_BLOCK_CALL       {$$ = TrCreateLeafNode (PARSEOP_ACCE
2166     | PARSEOP_ACCESSATTRIB_BYTE             {$$ = TrCreateLeafNode (PARSEOP_ACCE
2167     | PARSEOP_ACCESSATTRIB_QUICK            {$$ = TrCreateLeafNode (PARSEOP_ACCE
2168     | PARSEOP_ACCESSATTRIB_SND_RCV          {$$ = TrCreateLeafNode (PARSEOP_ACCE
2169     | PARSEOP_ACCESSATTRIB_WORD             {$$ = TrCreateLeafNode (PARSEOP_ACCE
2170     | PARSEOP_ACCESSATTRIB_WORD_CALL        {$$ = TrCreateLeafNode (PARSEOP_ACCE
2171     | PARSEOP_ACCESSATTRIB_MULTIBYTE '('    {$<n>$ = TrCreateLeafNode (PARSEOP_A
2172         ByteConst
```



```
2173         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
2174     | PARSEOP_ACCESSATTRIB_RAW_BYTES '('    {$<n>$ = TrCreateLeafNode (PARSEOP_A
2175         ByteConst
2176         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
2177     | PARSEOP_ACCESSATTRIB_RAW_PROCESS '('  {$<n>$ = TrCreateLeafNode (PARSEOP_A
2178         ByteConst
2179         ')'                        {$$ = TrLinkChildren ($<n>3,1,$4);}
2180     ;

2182 AccessTypeKeyword
2183     : PARSEOP_ACCESSTYPE_ANY                {$$ = TrCreateLeafNode (PARSEOP_ACCE
2184     | PARSEOP_ACCESSTYPE_BYTE               {$$ = TrCreateLeafNode (PARSEOP_ACCE
2185     | PARSEOP_ACCESSTYPE_WORD               {$$ = TrCreateLeafNode (PARSEOP_ACCE
2186     | PARSEOP_ACCESSTYPE_DWORD              {$$ = TrCreateLeafNode (PARSEOP_ACCE
2187     | PARSEOP_ACCESSTYPE_QWORD              {$$ = TrCreateLeafNode (PARSEOP_ACCE
2188     | PARSEOP_ACCESSTYPE_BUF                {$$ = TrCreateLeafNode (PARSEOP_ACCE
2189     ;

2191 AddressingModeKeyword
2192     : PARSEOP_ADDRESSINGMODE_7BIT           {$$ = TrCreateLeafNode (PARSEOP_ADDR
2193     | PARSEOP_ADDRESSINGMODE_10BIT          {$$ = TrCreateLeafNode (PARSEOP_ADDR
2194     ;

2196 AddressKeyword
2197     : PARSEOP_ADDRESSTYPE_MEMORY            {$$ = TrCreateLeafNode (PARSEOP_ADDR
2198     | PARSEOP_ADDRESSTYPE_RESERVED          {$$ = TrCreateLeafNode (PARSEOP_ADDR
2199     | PARSEOP_ADDRESSTYPE_NVS               {$$ = TrCreateLeafNode (PARSEOP_ADDR
2200     | PARSEOP_ADDRESSTYPE_ACPI              {$$ = TrCreateLeafNode (PARSEOP_ADDR
2201     ;

2203 AddressSpaceKeyword
2204     : ByteConst                             {$$ = UtCheckIntegerRange ($1, 0x0A,
2205     | RegionSpaceKeyword                    {}
2206     ;

2208 BitsPerByteKeyword
2209     : PARSEOP_BITSPERBYTE_FIVE              {$$ = TrCreateLeafNode (PARSEOP_BITS
2210     | PARSEOP_BITSPERBYTE_SIX               {$$ = TrCreateLeafNode (PARSEOP_BITS
2211     | PARSEOP_BITSPERBYTE_SEVEN             {$$ = TrCreateLeafNode (PARSEOP_BITS
2212     | PARSEOP_BITSPERBYTE_EIGHT             {$$ = TrCreateLeafNode (PARSEOP_BITS
2213     | PARSEOP_BITSPERBYTE_NINE              {$$ = TrCreateLeafNode (PARSEOP_BITS
2214     ;

2216 ClockPhaseKeyword
2217     : PARSEOP_CLOCKPHASE_FIRST              {$$ = TrCreateLeafNode (PARSEOP_CLOC
2218     | PARSEOP_CLOCKPHASE_SECOND             {$$ = TrCreateLeafNode (PARSEOP_CLOC
2219     ;

2221 ClockPolarityKeyword
2222     : PARSEOP_CLOCKPOLARITY_LOW             {$$ = TrCreateLeafNode (PARSEOP_CLOC
2223     | PARSEOP_CLOCKPOLARITY_HIGH            {$$ = TrCreateLeafNode (PARSEOP_CLOC
2224     ;

2226 DecodeKeyword
2227     : PARSEOP_DECODETYPE_POS                {$$ = TrCreateLeafNode (PARSEOP_DECO
2228     | PARSEOP_DECODETYPE_SUB                {$$ = TrCreateLeafNode (PARSEOP_DECO
2229     ;

2231 DevicePolarityKeyword
2232     : PARSEOP_DEVICEPOLARITY_LOW            {$$ = TrCreateLeafNode (PARSEOP_DEVI
2233     | PARSEOP_DEVICEPOLARITY_HIGH           {$$ = TrCreateLeafNode (PARSEOP_DEVI
2234     ;

2236 DMATypeKeyword
2237     : PARSEOP_DMATYPE_A                     {$$ = TrCreateLeafNode (PARSEOP_DMAT
2238     | PARSEOP_DMATYPE_COMPATIBILITY         {$$ = TrCreateLeafNode (PARSEOP_DMAT
```

```
2239     | PARSEOP_DMATYPE_B                 {$$ = TrCreateLeafNode (PARSEOP_DMAT
2240     | PARSEOP_DMATYPE_F                 {$$ = TrCreateLeafNode (PARSEOP_DMAT
2241     ;

2243 EndianKeyword
2244     : PARSEOP_ENDIAN_LITTLE             {$$ = TrCreateLeafNode (PARSEOP_ENDI
2245     | PARSEOP_ENDIAN_BIG                {$$ = TrCreateLeafNode (PARSEOP_ENDI
2246     ;

2248 FlowControlKeyword
2249     : PARSEOP_FLOWCONTROL_HW            {$$ = TrCreateLeafNode (PARSEOP_FLOW
2250     | PARSEOP_FLOWCONTROL_NONE          {$$ = TrCreateLeafNode (PARSEOP_FLOW
2251     | PARSEOP_FLOWCONTROL_SW            {$$ = TrCreateLeafNode (PARSEOP_FLOW
2252     ;

2254 InterruptLevel
2255     : PARSEOP_INTLEVEL_ACTIVEBOTH       {$$ = TrCreateLeafNode (PARSEOP_INTL
2256     | PARSEOP_INTLEVEL_ACTIVEHIGH       {$$ = TrCreateLeafNode (PARSEOP_INTL
2257     | PARSEOP_INTLEVEL_ACTIVELOW        {$$ = TrCreateLeafNode (PARSEOP_INTL
2258     ;

2260 InterruptTypeKeyword
2261     : PARSEOP_INTTYPE_EDGE              {$$ = TrCreateLeafNode (PARSEOP_INTT
2262     | PARSEOP_INTTYPE_LEVEL             {$$ = TrCreateLeafNode (PARSEOP_INTT
2263     ;

2265 IODecodeKeyword
2266     : PARSEOP_IODECODETYPE_16           {$$ = TrCreateLeafNode (PARSEOP_IODE
2267     | PARSEOP_IODECODETYPE_10           {$$ = TrCreateLeafNode (PARSEOP_IODE
2268     ;

2270 IoRestrictionKeyword
2271     : PARSEOP_IORESTRICT_IN             {$$ = TrCreateLeafNode (PARSEOP_IORE
2272     | PARSEOP_IORESTRICT_OUT            {$$ = TrCreateLeafNode (PARSEOP_IORE
2273     | PARSEOP_IORESTRICT_NONE           {$$ = TrCreateLeafNode (PARSEOP_IORE
2274     | PARSEOP_IORESTRICT_PRESERVE       {$$ = TrCreateLeafNode (PARSEOP_IORE
2275     ;

2277 LockRuleKeyword
2278     : PARSEOP_LOCKRULE_LOCK             {$$ = TrCreateLeafNode (PARSEOP_LOCK
2279     | PARSEOP_LOCKRULE_NOLOCK           {$$ = TrCreateLeafNode (PARSEOP_LOCK
2280     ;

2282 MatchOpKeyword
2283     : PARSEOP_MATCHTYPE_MTR             {$$ = TrCreateLeafNode (PARSEOP_MATC
2284     | PARSEOP_MATCHTYPE_MEQ             {$$ = TrCreateLeafNode (PARSEOP_MATC
2285     | PARSEOP_MATCHTYPE_MLE             {$$ = TrCreateLeafNode (PARSEOP_MATC
2286     | PARSEOP_MATCHTYPE_MLT             {$$ = TrCreateLeafNode (PARSEOP_MATC
2287     | PARSEOP_MATCHTYPE_MGE             {$$ = TrCreateLeafNode (PARSEOP_MATC
2288     | PARSEOP_MATCHTYPE_MGT             {$$ = TrCreateLeafNode (PARSEOP_MATC
2289     ;

2291 MaxKeyword
2292     : PARSEOP_MAXTYPE_FIXED             {$$ = TrCreateLeafNode (PARSEOP_MAXT
2293     | PARSEOP_MAXTYPE_NOTFIXED          {$$ = TrCreateLeafNode (PARSEOP_MAXT
2294     ;

2296 MemTypeKeyword
2297     : PARSEOP_MEMTYPE_CACHEABLE         {$$ = TrCreateLeafNode (PARSEOP_MEMT
2298     | PARSEOP_MEMTYPE_WRITECOMBINING    {$$ = TrCreateLeafNode (PARSEOP_MEMT
2299     | PARSEOP_MEMTYPE_PREFETCHABLE      {$$ = TrCreateLeafNode (PARSEOP_MEMT
2300     | PARSEOP_MEMTYPE_NONCACHEABLE      {$$ = TrCreateLeafNode (PARSEOP_MEMT
2301     ;

2303 MinKeyword
2304     : PARSEOP_MINTYPE_FIXED             {$$ = TrCreateLeafNode (PARSEOP_MINT
```



```
2305     | PARSEOP_MINTYPE_NOTFIXED          {$$ = TrCreateLeafNode (PARSEOP_MINT
2306     ;

2308 ObjectTypeKeyword
2309     : PARSEOP_OBJECTTYPE_UNK            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2310     | PARSEOP_OBJECTTYPE_INT            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2311     | PARSEOP_OBJECTTYPE_STR            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2312     | PARSEOP_OBJECTTYPE_BUF            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2313     | PARSEOP_OBJECTTYPE_PKG            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2314     | PARSEOP_OBJECTTYPE_FLD            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2315     | PARSEOP_OBJECTTYPE_DEV            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2316     | PARSEOP_OBJECTTYPE_EVT            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2317     | PARSEOP_OBJECTTYPE_MTH            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2318     | PARSEOP_OBJECTTYPE_MTX            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2319     | PARSEOP_OBJECTTYPE_OPR            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2320     | PARSEOP_OBJECTTYPE_POW            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2321     | PARSEOP_OBJECTTYPE_PRO            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2322     | PARSEOP_OBJECTTYPE_THZ            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2323     | PARSEOP_OBJECTTYPE_BFF            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2324     | PARSEOP_OBJECTTYPE_DDB            {$$ = TrCreateLeafNode (PARSEOP_OBJE
2325     ;

2327 ParityTypeKeyword
2328     : PARSEOP_PARITYTYPE_SPACE          {$$ = TrCreateLeafNode (PARSEOP_PARI
2329     | PARSEOP_PARITYTYPE_MARK           {$$ = TrCreateLeafNode (PARSEOP_PARI
2330     | PARSEOP_PARITYTYPE_ODD            {$$ = TrCreateLeafNode (PARSEOP_PARI
2331     | PARSEOP_PARITYTYPE_EVEN           {$$ = TrCreateLeafNode (PARSEOP_PARI
2332     | PARSEOP_PARITYTYPE_NONE           {$$ = TrCreateLeafNode (PARSEOP_PARI
2333     ;

2335 PinConfigByte
2336     : PinConfigKeyword                  {$$ = $1;}
2337     | ByteConstExpr                     {$$ = UtCheckIntegerRange ($1, 0x80,
2338     ;

2340 PinConfigKeyword
2341     : PARSEOP_PIN_NOPULL                {$$ = TrCreateLeafNode (PARSEOP_PIN_
2342     | PARSEOP_PIN_PULLDOWN              {$$ = TrCreateLeafNode (PARSEOP_PIN_
2343     | PARSEOP_PIN_PULLUP                {$$ = TrCreateLeafNode (PARSEOP_PIN_
2344     | PARSEOP_PIN_PULLDEFAULT           {$$ = TrCreateLeafNode (PARSEOP_PIN_
2345     ;

2347 RangeTypeKeyword
2348     : PARSEOP_RANGETYPE_ISAONLY         {$$ = TrCreateLeafNode (PARSEOP_RANG
2349     | PARSEOP_RANGETYPE_NONISAONLY      {$$ = TrCreateLeafNode (PARSEOP_RANG
2350     | PARSEOP_RANGETYPE_ENTIRE          {$$ = TrCreateLeafNode (PARSEOP_RANG
2351     ;

2353 RegionSpaceKeyword
2354     : PARSEOP_REGIONSPACE_IO            {$$ = TrCreateLeafNode (PARSEOP_REGI
2355     | PARSEOP_REGIONSPACE_MEM           {$$ = TrCreateLeafNode (PARSEOP_REGI
2356     | PARSEOP_REGIONSPACE_PCI           {$$ = TrCreateLeafNode (PARSEOP_REGI
2357     | PARSEOP_REGIONSPACE_EC            {$$ = TrCreateLeafNode (PARSEOP_REGI
2358     | PARSEOP_REGIONSPACE_SMBUS         {$$ = TrCreateLeafNode (PARSEOP_REGI
2359     | PARSEOP_REGIONSPACE_CMOS          {$$ = TrCreateLeafNode (PARSEOP_REGI
2360     | PARSEOP_REGIONSPACE_PCIBAR        {$$ = TrCreateLeafNode (PARSEOP_REGI
2361     | PARSEOP_REGIONSPACE_IPMI          {$$ = TrCreateLeafNode (PARSEOP_REGI
2362     | PARSEOP_REGIONSPACE_GPIO          {$$ = TrCreateLeafNode (PARSEOP_REGI
2363     | PARSEOP_REGIONSPACE_GSBUS         {$$ = TrCreateLeafNode (PARSEOP_REGI
2364     | PARSEOP_REGIONSPACE_PCC           {$$ = TrCreateLeafNode (PARSEOP_REGI
2365     | PARSEOP_REGIONSPACE_FFIXEDHW      {$$ = TrCreateLeafNode (PARSEOP_REGI
2366     ;

2368 ResourceTypeKeyword
2369     : PARSEOP_RESOURCETYPE_CONSUMER     {$$ = TrCreateLeafNode (PARSEOP_RESO
2370     | PARSEOP_RESOURCETYPE_PRODUCER     {$$ = TrCreateLeafNode (PARSEOP_RESO
```

```
2371     ;

2373 SerializeRuleKeyword
2374     : PARSEOP_SERIALIZERULE_SERIAL          {$$ = TrCreateLeafNode (PARSEOP_SERI
2375     | PARSEOP_SERIALIZERULE_NOTSERIAL       {$$ = TrCreateLeafNode (PARSEOP_SERI
2376     ;

2378 ShareTypeKeyword
2379     : PARSEOP_SHARETYPE_SHARED              {$$ = TrCreateLeafNode (PARSEOP_SHAR
2380     | PARSEOP_SHARETYPE_EXCLUSIVE           {$$ = TrCreateLeafNode (PARSEOP_SHAR
2381     | PARSEOP_SHARETYPE_SHAREDWAKE          {$$ = TrCreateLeafNode (PARSEOP_SHAR
2382     | PARSEOP_SHARETYPE_EXCLUSIVEWAKE       {$$ = TrCreateLeafNode (PARSEOP_SHAR
2383    ;

2385 SlaveModeKeyword
2386     : PARSEOP_SLAVEMODE_CONTROLLERINIT      {$$ = TrCreateLeafNode (PARSEOP_SLAV
2387     | PARSEOP_SLAVEMODE_DEVICEINIT          {$$ = TrCreateLeafNode (PARSEOP_SLAV
2388     ;

2390 StopBitsKeyword
2391     : PARSEOP_STOPBITS_TWO                  {$$ = TrCreateLeafNode (PARSEOP_STOP
2392     | PARSEOP_STOPBITS_ONEPLUSHALF          {$$ = TrCreateLeafNode (PARSEOP_STOP
2393     | PARSEOP_STOPBITS_ONE                  {$$ = TrCreateLeafNode (PARSEOP_STOP
2394     | PARSEOP_STOPBITS_ZERO                 {$$ = TrCreateLeafNode (PARSEOP_STOP
2395     ;

2397 TranslationKeyword
2398     : PARSEOP_TRANSLATIONTYPE_SPARSE        {$$ = TrCreateLeafNode (PARSEOP_TRAN
2399     | PARSEOP_TRANSLATIONTYPE_DENSE         {$$ = TrCreateLeafNode (PARSEOP_TRAN
2400     ;

2402 TypeKeyword
2403     : PARSEOP_TYPE_TRANSLATION              {$$ = TrCreateLeafNode (PARSEOP_TYPE
2404     | PARSEOP_TYPE_STATIC                   {$$ = TrCreateLeafNode (PARSEOP_TYPE
2405     ;

2407 UpdateRuleKeyword
2408     : PARSEOP_UPDATERULE_PRESERVE           {$$ = TrCreateLeafNode (PARSEOP_UPDA
2409     | PARSEOP_UPDATERULE_ONES               {$$ = TrCreateLeafNode (PARSEOP_UPDA
2410     | PARSEOP_UPDATERULE_ZEROS              {$$ = TrCreateLeafNode (PARSEOP_UPDA
2411     ;

2413 WireModeKeyword
2414     : PARSEOP_WIREMODE_FOUR                 {$$ = TrCreateLeafNode (PARSEOP_WIRE
2415     | PARSEOP_WIREMODE_THREE                {$$ = TrCreateLeafNode (PARSEOP_WIRE
2416     ;

2418 XferSizeKeyword
2419     : PARSEOP_XFERSIZE_8                    {$$ = TrCreateValuedLeafNode (PARSEO
2420     | PARSEOP_XFERSIZE_16                   {$$ = TrCreateValuedLeafNode (PARSEO
2421     | PARSEOP_XFERSIZE_32                   {$$ = TrCreateValuedLeafNode (PARSEO
2422     | PARSEOP_XFERSIZE_64                   {$$ = TrCreateValuedLeafNode (PARSEO
2423     | PARSEOP_XFERSIZE_128                  {$$ = TrCreateValuedLeafNode (PARSEO
2424     | PARSEOP_XFERSIZE_256                  {$$ = TrCreateValuedLeafNode (PARSEO
2425     ;

2427 XferTypeKeyword
2428     : PARSEOP_XFERTYPE_8                    {$$ = TrCreateLeafNode (PARSEOP_XFER
2429     | PARSEOP_XFERTYPE_8_16                 {$$ = TrCreateLeafNode (PARSEOP_XFER
2430     | PARSEOP_XFERTYPE_16                   {$$ = TrCreateLeafNode (PARSEOP_XFER
2431     ;


2434 /******* Miscellaneous Types ************************************************/
```



```
2437 SuperName
2438     : NameString                    {}
2439     | ArgTerm                       {}
2440     | LocalTerm                     {}
2441     | DebugTerm                     {}
2442     | Type6Opcode                   {}

2444 /* For ObjectType: SuperName except for UserTerm (method invocation) */

2446 ObjectTypeName
2447     : NameString                    {}
2448     | ArgTerm                       {}
2449     | LocalTerm                     {}
2450     | DebugTerm                     {}
2451     | RefOfTerm                     {}
2452     | DerefOfTerm                   {}
2453     | IndexTerm                     {}

2455 /*    | UserTerm                      {} */  /* Caused reduce/reduce with Type6O
2456     ;

2458 ArgTerm
2459     : PARSEOP_ARG0                  {$$ = TrCreateLeafNode (PARSEOP_ARG0);}
2460     | PARSEOP_ARG1                  {$$ = TrCreateLeafNode (PARSEOP_ARG1);}
2461     | PARSEOP_ARG2                  {$$ = TrCreateLeafNode (PARSEOP_ARG2);}
2462     | PARSEOP_ARG3                  {$$ = TrCreateLeafNode (PARSEOP_ARG3);}
2463     | PARSEOP_ARG4                  {$$ = TrCreateLeafNode (PARSEOP_ARG4);}
2464     | PARSEOP_ARG5                  {$$ = TrCreateLeafNode (PARSEOP_ARG5);}
2465     | PARSEOP_ARG6                  {$$ = TrCreateLeafNode (PARSEOP_ARG6);}
2466     ;

2468 LocalTerm
2469     : PARSEOP_LOCAL0                {$$ = TrCreateLeafNode (PARSEOP_LOCAL0);}
2470     | PARSEOP_LOCAL1                {$$ = TrCreateLeafNode (PARSEOP_LOCAL1);}
2471     | PARSEOP_LOCAL2                {$$ = TrCreateLeafNode (PARSEOP_LOCAL2);}
2472     | PARSEOP_LOCAL3                {$$ = TrCreateLeafNode (PARSEOP_LOCAL3);}
2473     | PARSEOP_LOCAL4                {$$ = TrCreateLeafNode (PARSEOP_LOCAL4);}
2474     | PARSEOP_LOCAL5                {$$ = TrCreateLeafNode (PARSEOP_LOCAL5);}
2475     | PARSEOP_LOCAL6                {$$ = TrCreateLeafNode (PARSEOP_LOCAL6);}
2476     | PARSEOP_LOCAL7                {$$ = TrCreateLeafNode (PARSEOP_LOCAL7);}
2477     ;

2479 DebugTerm
2480     : PARSEOP_DEBUG                 {$$ = TrCreateLeafNode (PARSEOP_DEBUG);}
2481     ;


2484 ByteConst
2485     : Integer                       {$$ = TrUpdateNode (PARSEOP_BYTECONST, $1);}
2486     ;

2488 WordConst
2489     : Integer                       {$$ = TrUpdateNode (PARSEOP_WORDCONST, $1);}
2490     ;

2492 DWordConst
2493     : Integer                       {$$ = TrUpdateNode (PARSEOP_DWORDCONST, $1);
2494     ;

2496 QWordConst
2497     : Integer                       {$$ = TrUpdateNode (PARSEOP_QWORDCONST, $1);
2498     ;

2500 Integer
2501     : PARSEOP_INTEGER               {$$ = TrCreateValuedLeafNode (PARSEOP_INTEGE
2502     ;
```

```
2504 String
2505     : PARSEOP_STRING_LITERAL        {$$ = TrCreateValuedLeafNode (PARSEOP_STRING
2506     ;

2508 ConstTerm
2509     : ConstExprTerm                 {}
2510     | PARSEOP_REVISION              {$$ = TrCreateLeafNode (PARSEOP_REVISION);}
2511     ;

2513 ConstExprTerm
2514     : PARSEOP_ZERO                  {$$ = TrCreateValuedLeafNode (PARSEOP_ZERO,
2515     | PARSEOP_ONE                   {$$ = TrCreateValuedLeafNode (PARSEOP_ONE, 1
2516     | PARSEOP_ONES                  {$$ = TrCreateValuedLeafNode (PARSEOP_ONES,
2517     | PARSEOP___DATE__              {$$ = TrCreateConstantLeafNode (PARSEOP___DA
2518     | PARSEOP___FILE__              {$$ = TrCreateConstantLeafNode (PARSEOP___FI
2519     | PARSEOP___LINE__              {$$ = TrCreateConstantLeafNode (PARSEOP___LI
2520     | PARSEOP___PATH__              {$$ = TrCreateConstantLeafNode (PARSEOP___PA
2521     ;

2523 /*
2524  * The NODE_COMPILE_TIME_CONST flag in the following constant expressions
2525  * enables compile-time constant folding to reduce the Type3Opcodes/Type2Integer
2526  * to simple integers. It is an error if these types of expressions cannot be
2527  * reduced, since the AML grammar for ****ConstExpr requires a simple constant.
2528  * Note: The required byte length of the constant is passed through to the
2529  * constant folding code in the node AmlLength field.
2530  */
2531 ByteConstExpr
2532     : Type3Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2533     | Type2IntegerOpcode            {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2534     | ConstExprTerm                 {$$ = TrUpdateNode (PARSEOP_BYTECONST, $1);}
2535     | ByteConst                     {}
2536     ;

2538 WordConstExpr
2539     : Type3Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2540     | Type2IntegerOpcode            {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2541     | ConstExprTerm                 {$$ = TrUpdateNode (PARSEOP_WORDCONST, $1);}
2542     | WordConst                     {}
2543     ;

2545 DWordConstExpr
2546     : Type3Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2547     | Type2IntegerOpcode            {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2548     | ConstExprTerm                 {$$ = TrUpdateNode (PARSEOP_DWORDCONST, $1);
2549     | DWordConst                    {}
2550     ;

2552 QWordConstExpr
2553     : Type3Opcode                   {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2554     | Type2IntegerOpcode            {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME_
2555     | ConstExprTerm                 {$$ = TrUpdateNode (PARSEOP_QWORDCONST, $1);
2556     | QWordConst                    {}
2557     ;

2559 /* OptionalCount must appear before ByteList or an incorrect reduction will resu

2561 OptionalCount
2562     :                               {$$ = TrCreateLeafNode (PARSEOP_ONES);}
2563     | ','                           {$$ = TrCreateLeafNode (PARSEOP_ONES);}
2564     | ',' TermArg                   {$$ = $2;}
2565     ;

2567 BufferTerm
2568     : PARSEOP_BUFFER '('            {$<n>$ = TrCreateLeafNode (PARSEOP_BUFFER);}
```



```
2569         OptionalTermArg
2570         ')' '{'
2571             BufferTermData '}'      {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
2572     | PARSEOP_BUFFER '('
2573         error ')'                   {$$ = AslDoError(); yyclearin;}
2574     ;

2576 BufferTermData
2577     : ByteList                      {}
2578     | StringData                    {}
2579     ;

2581 ByteList
2582     :                               {$$ = NULL;}
2583     | ByteConstExpr
2584     | ByteList ','                  /* Allows a trailing comma at list end */
2585     | ByteList ','
2586         ByteConstExpr               {$$ = TrLinkPeerNode ($1,$3);}
2587     ;

2589 DataBufferTerm
2590     : PARSEOP_DATABUFFER  '('       {$<n>$ = TrCreateLeafNode (PARSEOP_DATABUFFE
2591         OptionalWordConst
2592         ')' '{'
2593             ByteList '}'            {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
2594     | PARSEOP_DATABUFFER '('
2595         error ')'                   {$$ = AslDoError(); yyclearin;}
2596     ;

2598 DWordList
2599     :                               {$$ = NULL;}
2600     | DWordConstExpr
2601     | DWordList ','                 /* Allows a trailing comma at list end */
2602     | DWordList ','
2603         DWordConstExpr              {$$ = TrLinkPeerNode ($1,$3);}
2604     ;

2606 PackageTerm
2607     : PARSEOP_PACKAGE '('           {$<n>$ = TrCreateLeafNode (PARSEOP_VAR_PACKA
2608         VarPackageLengthTerm
2609         ')' '{'
2610             PackageList '}'         {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
2611     | PARSEOP_PACKAGE '('
2612         error ')'                   {$$ = AslDoError(); yyclearin;}
2613     ;

2615 PackageList
2616     :                               {$$ = NULL;}
2617     | PackageElement
2618     | PackageList ','               /* Allows a trailing comma at list end */
2619     | PackageList ','
2620         PackageElement              {$$ = TrLinkPeerNode ($1,$3);}
2621     ;

2623 PackageElement
2624     : DataObject                    {}
2625     | NameString                    {}
2626     ;

2628 VarPackageLengthTerm
2629     :                               {$$ = TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
2630     | TermArg                       {$$ = $1;}
2631     ;


2634 /******* Macros *********************************************/
```

```
2637 EISAIDTerm
2638     : PARSEOP_EISAID '('
2639         StringData ')'              {$$ = TrUpdateNode (PARSEOP_EISAID, $3);}
2640     | PARSEOP_EISAID '('
2641         error ')'                   {$$ = AslDoError(); yyclearin;}
2642     ;

2644 UnicodeTerm
2645     : PARSEOP_UNICODE '('           {$<n>$ = TrCreateLeafNode (PARSEOP_UNICODE);
2646         StringData
2647         ')'                         {$$ = TrLinkChildren ($<n>3,2,0,$4);}
2648     | PARSEOP_UNICODE '('
2649         error ')'                   {$$ = AslDoError(); yyclearin;}
2650     ;


2653 /******* Resources and Memory ***********************************************/


2656 /*
2657  * Note: Create two default nodes to allow conversion to a Buffer AML opcode
2658  * Also, insert the EndTag at the end of the template.
2659  */
2660 ResourceTemplateTerm
2661     : PARSEOP_RESOURCETEMPLATE '(' ')'
2662         '{'
2663         ResourceMacroList '}'       {$$ = TrCreateNode (PARSEOP_RESOURCETEMPLATE
2664                                           TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
2665                                           TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
2666                                           $5,
2667                                           TrCreateLeafNode (PARSEOP_ENDTAG));}
2668     ;

2670 ResourceMacroList
2671     :                               {$$ = NULL;}
2672     | ResourceMacroList
2673         ResourceMacroTerm           {$$ = TrLinkPeerNode ($1,$2);}
2674     ;

2676 ResourceMacroTerm
2677     : DMATerm                       {}
2678     | DWordIOTerm                   {}
2679     | DWordMemoryTerm               {}
2680     | DWordSpaceTerm                {}
2681     | EndDependentFnTerm            {}
2682     | ExtendedIOTerm                {}
2683     | ExtendedMemoryTerm            {}
2684     | ExtendedSpaceTerm             {}
2685     | FixedDmaTerm                  {}
2686     | FixedIOTerm                   {}
2687     | GpioIntTerm                   {}
2688     | GpioIoTerm                    {}
2689     | I2cSerialBusTerm              {}
2690     | InterruptTerm                 {}
2691     | IOTerm                        {}
2692     | IRQNoFlagsTerm                {}
2693     | IRQTerm                       {}
2694     | Memory24Term                  {}
2695     | Memory32FixedTerm             {}
2696     | Memory32Term                  {}
2697     | QWordIOTerm                   {}
2698     | QWordMemoryTerm               {}
2699     | QWordSpaceTerm                {}
2700     | RegisterTerm                  {}
```



```
2701     | SpiSerialBusTerm              {}
2702     | StartDependentFnNoPriTerm     {}
2703     | StartDependentFnTerm          {}
2704     | UartSerialBusTerm             {}
2705     | VendorLongTerm                {}
2706     | VendorShortTerm               {}
2707     | WordBusNumberTerm             {}
2708     | WordIOTerm                    {}
2709     | WordSpaceTerm                 {}
2710     ;

2712 DMATerm
2713     : PARSEOP_DMA '('               {$<n>$ = TrCreateLeafNode (PARSEOP_DMA);}
2714         DMATypeKeyword
2715         OptionalBusMasterKeyword
2716         ',' XferTypeKeyword
2717         OptionalNameString_Last
2718         ')' '{'
2719             ByteList '}'            {$$ = TrLinkChildren ($<n>3,5,$4,$5,$7,$8,$1
2720     | PARSEOP_DMA '('
2721         error ')'                   {$$ = AslDoError(); yyclearin;}
2722     ;

2724 DWordIOTerm
2725     : PARSEOP_DWORDIO '('           {$<n>$ = TrCreateLeafNode (PARSEOP_DWORDIO);
2726         OptionalResourceType_First
2727         OptionalMinType
2728         OptionalMaxType
2729         OptionalDecodeType
2730         OptionalRangeType
2731         ',' DWordConstExpr
2732         ',' DWordConstExpr
2733         ',' DWordConstExpr
2734         ',' DWordConstExpr
2735         ',' DWordConstExpr
2736         OptionalByteConstExpr
2737         OptionalStringData
2738         OptionalNameString
2739         OptionalType
2740         OptionalTranslationType_Last
2741         ')'                         {$$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
2742     | PARSEOP_DWORDIO '('
2743         error ')'                   {$$ = AslDoError(); yyclearin;}
2744     ;

2746 DWordMemoryTerm
2747     : PARSEOP_DWORDMEMORY '('       {$<n>$ = TrCreateLeafNode (PARSEOP_DWORDMEMO
2748         OptionalResourceType_First
2749         OptionalDecodeType
2750         OptionalMinType
2751         OptionalMaxType
2752         OptionalMemType
2753         ',' OptionalReadWriteKeyword
2754         ',' DWordConstExpr
2755         ',' DWordConstExpr
2756         ',' DWordConstExpr
2757         ',' DWordConstExpr
2758         ',' DWordConstExpr
2759         OptionalByteConstExpr
2760         OptionalStringData
2761         OptionalNameString
2762         OptionalAddressRange
2763         OptionalType_Last
2764         ')'                         {$$ = TrLinkChildren ($<n>3,16,$4,$5,$6,$7,$
2765     | PARSEOP_DWORDMEMORY '('
2766         error ')'                   {$$ = AslDoError(); yyclearin;}
```

```
2767     ;

2769 DWordSpaceTerm
2770     : PARSEOP_DWORDSPACE '('        {$<n>$ = TrCreateLeafNode (PARSEOP_DWORDSPAC
2771         ByteConstExpr               {UtCheckIntegerRange ($4, 0xC0, 0xFF);}
2772         OptionalResourceType
2773         OptionalDecodeType
2774         OptionalMinType
2775         OptionalMaxType
2776         ',' ByteConstExpr
2777         ',' DWordConstExpr
2778         ',' DWordConstExpr
2779         ',' DWordConstExpr
2780         ',' DWordConstExpr
2781         ',' DWordConstExpr
2782         OptionalByteConstExpr
2783         OptionalStringData
2784         OptionalNameString_Last
2785         ')'                         {$$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
2786     | PARSEOP_DWORDSPACE '('
2787         error ')'                   {$$ = AslDoError(); yyclearin;}
2788     ;


2791 EndDependentFnTerm
2792     : PARSEOP_ENDDEPENDENTFN '('
2793         ')'                         {$$ = TrCreateLeafNode (PARSEOP_ENDDEPENDENT
2794     | PARSEOP_ENDDEPENDENTFN '('
2795         error ')'                   {$$ = AslDoError(); yyclearin;}
2796     ;

2798 ExtendedIOTerm
2799     : PARSEOP_EXTENDEDIO '('        {$<n>$ = TrCreateLeafNode (PARSEOP_EXTENDEDI
2800         OptionalResourceType_First
2801         OptionalMinType
2802         OptionalMaxType
2803         OptionalDecodeType
2804         OptionalRangeType
2805         ',' QWordConstExpr
2806         ',' QWordConstExpr
2807         ',' QWordConstExpr
2808         ',' QWordConstExpr
2809         ',' QWordConstExpr
2810         OptionalQWordConstExpr
2811         OptionalNameString
2812         OptionalType
2813         OptionalTranslationType_Last
2814         ')'                         {$$ = TrLinkChildren ($<n>3,14,$4,$5,$6,$7,$
2815     | PARSEOP_EXTENDEDIO '('
2816         error ')'                   {$$ = AslDoError(); yyclearin;}
2817     ;

2819 ExtendedMemoryTerm
2820     : PARSEOP_EXTENDEDMEMORY '('    {$<n>$ = TrCreateLeafNode (PARSEOP_EXTENDEDM
2821         OptionalResourceType_First
2822         OptionalDecodeType
2823         OptionalMinType
2824         OptionalMaxType
2825         OptionalMemType
2826         ',' OptionalReadWriteKeyword
2827         ',' QWordConstExpr
2828         ',' QWordConstExpr
2829         ',' QWordConstExpr
2830         ',' QWordConstExpr
2831         ',' QWordConstExpr
2832         OptionalQWordConstExpr
```



```
2833         OptionalNameString
2834         OptionalAddressRange
2835         OptionalType_Last
2836         ')'                         {$$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
2837     | PARSEOP_EXTENDEDMEMORY '('
2838         error ')'                   {$$ = AslDoError(); yyclearin;}
2839     ;

2841 ExtendedSpaceTerm
2842     : PARSEOP_EXTENDEDSPACE '('     {$<n>$ = TrCreateLeafNode (PARSEOP_EXTENDEDS
2843         ByteConstExpr               {UtCheckIntegerRange ($4, 0xC0, 0xFF);}
2844         OptionalResourceType
2845         OptionalDecodeType
2846         OptionalMinType
2847         OptionalMaxType
2848         ',' ByteConstExpr
2849         ',' QWordConstExpr
2850         ',' QWordConstExpr
2851         ',' QWordConstExpr
2852         ',' QWordConstExpr
2853         ',' QWordConstExpr
2854         OptionalQWordConstExpr
2855         OptionalNameString_Last
2856         ')'                         {$$ = TrLinkChildren ($<n>3,13,$4,$6,$7,$8,$
2857     | PARSEOP_EXTENDEDSPACE '('
2858         error ')'                   {$$ = AslDoError(); yyclearin;}
2859     ;

2861 FixedDmaTerm
2862     : PARSEOP_FIXEDDMA '('          {$<n>$ = TrCreateLeafNode (PARSEOP_FIXEDDMA)
2863         WordConstExpr               /* 04: DMA RequestLines */
2864         ',' WordConstExpr           /* 06: DMA Channels */
2865         OptionalXferSize            /* 07: DMA TransferSize */
2866         OptionalNameString          /* 08: DescriptorName */
2867         ')'                         {$$ = TrLinkChildren ($<n>3,4,$4,$6,$7,$8);}
2868     | PARSEOP_FIXEDDMA '('
2869         error ')'                   {$$ = AslDoError(); yyclearin;}
2870     ;

2872 FixedIOTerm
2873     : PARSEOP_FIXEDIO '('           {$<n>$ = TrCreateLeafNode (PARSEOP_FIXEDIO);
2874         WordConstExpr
2875         ',' ByteConstExpr
2876         OptionalNameString_Last
2877         ')'                         {$$ = TrLinkChildren ($<n>3,3,$4,$6,$7);}
2878     | PARSEOP_FIXEDIO '('
2879         error ')'                   {$$ = AslDoError(); yyclearin;}
2880     ;

2882 GpioIntTerm
2883     : PARSEOP_GPIO_INT '('          {$<n>$ = TrCreateLeafNode (PARSEOP_GPIO_INT)
2884         InterruptTypeKeyword        /* 04: InterruptType */
2885         ',' InterruptLevel          /* 06: InterruptLevel */
2886         OptionalShareType           /* 07: SharedType */
2887         ',' PinConfigByte           /* 09: PinConfig */
2888         OptionalWordConstExpr       /* 10: DebounceTimeout */
2889         ',' StringData              /* 12: ResourceSource */
2890         OptionalByteConstExpr       /* 13: ResourceSourceIndex */
2891         OptionalResourceType        /* 14: ResourceType */
2892         OptionalNameString          /* 15: DescriptorName */
2893         OptionalBuffer_Last         /* 16: VendorData */
2894         ')' '{'
2895             DWordConstExpr '}'      {$$ = TrLinkChildren ($<n>3,11,$4,$6,$7,$9,$
2896     | PARSEOP_GPIO_INT '('
2897         error ')'                   {$$ = AslDoError(); yyclearin;}
2898     ;
```

```
2900 GpioIoTerm
2901     : PARSEOP_GPIO_IO '('          {$<n>$ = TrCreateLeafNode (PARSEOP_GPIO_IO);
2902         OptionalShareType_First    /* 04: SharedType */
2903         ',' PinConfigByte          /* 06: PinConfig */
2904         OptionalWordConstExpr      /* 07: DebounceTimeout */
2905         OptionalWordConstExpr      /* 08: DriveStrength */
2906         OptionalIoRestriction      /* 09: IoRestriction */
2907         ',' StringData             /* 11: ResourceSource */
2908         OptionalByteConstExpr      /* 12: ResourceSourceIndex */
2909         OptionalResourceType       /* 13: ResourceType */
2910         OptionalNameString         /* 14: DescriptorName */
2911         OptionalBuffer_Last        /* 15: VendorData */
2912         ')' '{'
2913             DWordList '}'          {$$ = TrLinkChildren ($<n>3,11,$4,$6,$7,$8,$
2914     | PARSEOP_GPIO_IO '('
2915         error ')'                  {$$ = AslDoError(); yyclearin;}
2916     ;

2918 I2cSerialBusTerm
2919     : PARSEOP_I2C_SERIALBUS '('    {$<n>$ = TrCreateLeafNode (PARSEOP_I2C_SERIA
2920         WordConstExpr              /* 04: SlaveAddress */
2921         OptionalSlaveMode          /* 05: SlaveMode */
2922         ',' DWordConstExpr         /* 07: ConnectionSpeed */
2923         OptionalAddressingMode     /* 08: AddressingMode */
2924         ',' StringData             /* 10: ResourceSource */
2925         OptionalByteConstExpr      /* 11: ResourceSourceIndex */
2926         OptionalResourceType       /* 12: ResourceType */
2927         OptionalNameString         /* 13: DescriptorName */
2928         OptionalBuffer_Last        /* 14: VendorData */
2929         ')'                        {$$ = TrLinkChildren ($<n>3,9,$4,$5,$7,$8,$1
2930     | PARSEOP_I2C_SERIALBUS '('
2931         error ')'                  {$$ = AslDoError(); yyclearin;}
2932     ;

2934 InterruptTerm
2935     : PARSEOP_INTERRUPT '('        {$<n>$ = TrCreateLeafNode (PARSEOP_INTERRUPT
2936         OptionalResourceType_First
2937         ',' InterruptTypeKeyword
2938         ',' InterruptLevel
2939         OptionalShareType
2940         OptionalByteConstExpr
2941         OptionalStringData
2942         OptionalNameString_Last
2943         ')' '{'
2944             DWordList '}'          {$$ = TrLinkChildren ($<n>3,8,$4,$6,$8,$9,$1
2945     | PARSEOP_INTERRUPT '('
2946         error ')'                  {$$ = AslDoError(); yyclearin;}
2947     ;

2949 IOTerm
2950     : PARSEOP_IO '('               {$<n>$ = TrCreateLeafNode (PARSEOP_IO);}
2951         IODecodeKeyword
2952         ',' WordConstExpr
2953         ',' WordConstExpr
2954         ',' ByteConstExpr
2955         ',' ByteConstExpr
2956         OptionalNameString_Last
2957         ')'                        {$$ = TrLinkChildren ($<n>3,6,$4,$6,$8,$10,$
2958     | PARSEOP_IO '('
2959         error ')'                  {$$ = AslDoError(); yyclearin;}
2960     ;

2962 IRQNoFlagsTerm
2963     : PARSEOP_IRQNOFLAGS '('       {$<n>$ = TrCreateLeafNode (PARSEOP_IRQNOFLAG
2964         OptionalNameString_First
```



```
2965         ')' '{'
2966             ByteList '}'           {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
2967     | PARSEOP_IRQNOFLAGS '('
2968         error ')'                  {$$ = AslDoError(); yyclearin;}
2969     ;

2971 IRQTerm
2972     : PARSEOP_IRQ '('              {$<n>$ = TrCreateLeafNode (PARSEOP_IRQ);}
2973         InterruptTypeKeyword
2974         ',' InterruptLevel
2975         OptionalShareType
2976         OptionalNameString_Last
2977         ')' '{'
2978             ByteList '}'           {$$ = TrLinkChildren ($<n>3,5,$4,$6,$7,$8,$1
2979     | PARSEOP_IRQ '('
2980         error ')'                  {$$ = AslDoError(); yyclearin;}
2981     ;

2983 Memory24Term
2984     : PARSEOP_MEMORY24 '('         {$<n>$ = TrCreateLeafNode (PARSEOP_MEMORY24)
2985         OptionalReadWriteKeyword
2986         ',' WordConstExpr
2987         ',' WordConstExpr
2988         ',' WordConstExpr
2989         ',' WordConstExpr
2990         OptionalNameString_Last
2991         ')'                        {$$ = TrLinkChildren ($<n>3,6,$4,$6,$8,$10,$
2992     | PARSEOP_MEMORY24 '('
2993         error ')'                  {$$ = AslDoError(); yyclearin;}
2994     ;

2996 Memory32FixedTerm
2997     : PARSEOP_MEMORY32FIXED '('    {$<n>$ = TrCreateLeafNode (PARSEOP_MEMORY32F
2998         OptionalReadWriteKeyword
2999         ',' DWordConstExpr
3000         ',' DWordConstExpr
3001         OptionalNameString_Last
3002         ')'                        {$$ = TrLinkChildren ($<n>3,4,$4,$6,$8,$9);}
3003     | PARSEOP_MEMORY32FIXED '('
3004         error ')'                  {$$ = AslDoError(); yyclearin;}
3005     ;

3007 Memory32Term
3008     : PARSEOP_MEMORY32 '('         {$<n>$ = TrCreateLeafNode (PARSEOP_MEMORY32)
3009         OptionalReadWriteKeyword
3010         ',' DWordConstExpr
3011         ',' DWordConstExpr
3012         ',' DWordConstExpr
3013         ',' DWordConstExpr
3014         OptionalNameString_Last
3015         ')'                        {$$ = TrLinkChildren ($<n>3,6,$4,$6,$8,$10,$
3016     | PARSEOP_MEMORY32 '('
3017         error ')'                  {$$ = AslDoError(); yyclearin;}
3018     ;

3020 QWordIOTerm
3021     : PARSEOP_QWORDIO '('          {$<n>$ = TrCreateLeafNode (PARSEOP_QWORDIO);
3022         OptionalResourceType_First
3023         OptionalMinType
3024         OptionalMaxType
3025         OptionalDecodeType
3026         OptionalRangeType
3027         ',' QWordConstExpr
3028         ',' QWordConstExpr
3029         ',' QWordConstExpr
3030         ',' QWordConstExpr
```

```
3031         ',' QWordConstExpr
3032         OptionalByteConstExpr
3033         OptionalStringData
3034         OptionalNameString
3035         OptionalType
3036         OptionalTranslationType_Last
3037         ')'                         {$$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
3038     | PARSEOP_QWORDIO '('
3039         error ')'                   {$$ = AslDoError(); yyclearin;}
3040     ;

3042 QWordMemoryTerm
3043     : PARSEOP_QWORDMEMORY '('       {$<n>$ = TrCreateLeafNode (PARSEOP_QWORDMEMO
3044         OptionalResourceType_First
3045         OptionalDecodeType
3046         OptionalMinType
3047         OptionalMaxType
3048         OptionalMemType
3049         ',' OptionalReadWriteKeyword
3050         ',' QWordConstExpr
3051         ',' QWordConstExpr
3052         ',' QWordConstExpr
3053         ',' QWordConstExpr
3054         ',' QWordConstExpr
3055         OptionalByteConstExpr
3056         OptionalStringData
3057         OptionalNameString
3058         OptionalAddressRange
3059         OptionalType_Last
3060         ')'                         {$$ = TrLinkChildren ($<n>3,16,$4,$5,$6,$7,$
3061     | PARSEOP_QWORDMEMORY '('
3062         error ')'                   {$$ = AslDoError(); yyclearin;}
3063     ;

3065 QWordSpaceTerm
3066     : PARSEOP_QWORDSPACE '('        {$<n>$ = TrCreateLeafNode (PARSEOP_QWORDSPAC
3067         ByteConstExpr               {UtCheckIntegerRange ($4, 0xC0, 0xFF);}
3068         OptionalResourceType
3069         OptionalDecodeType
3070         OptionalMinType
3071         OptionalMaxType
3072         ',' ByteConstExpr
3073         ',' QWordConstExpr
3074         ',' QWordConstExpr
3075         ',' QWordConstExpr
3076         ',' QWordConstExpr
3077         ',' QWordConstExpr
3078         OptionalByteConstExpr
3079         OptionalStringData
3080         OptionalNameString_Last
3081         ')'                         {$$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
3082     | PARSEOP_QWORDSPACE '('
3083         error ')'                   {$$ = AslDoError(); yyclearin;}
3084     ;

3086 RegisterTerm
3087     : PARSEOP_REGISTER '('          {$<n>$ = TrCreateLeafNode (PARSEOP_REGISTER)
3088         AddressSpaceKeyword
3089         ',' ByteConstExpr
3090         ',' ByteConstExpr
3091         ',' QWordConstExpr
3092         OptionalAccessSize
3093         OptionalNameString_Last
3094         ')'                         {$$ = TrLinkChildren ($<n>3,6,$4,$6,$8,$10,$
3095     | PARSEOP_REGISTER '('
3096         error ')'                   {$$ = AslDoError(); yyclearin;}
```



```
3097     ;

3099 SpiSerialBusTerm
3100     : PARSEOP_SPI_SERIALBUS '('     {$<n>$ = TrCreateLeafNode (PARSEOP_SPI_SERIA
3101         WordConstExpr               /* 04: DeviceSelection */
3102         OptionalDevicePolarity      /* 05: DevicePolarity */
3103         OptionalWireMode            /* 06: WireMode */
3104         ',' ByteConstExpr           /* 08: DataBitLength */
3105         OptionalSlaveMode           /* 09: SlaveMode */
3106         ',' DWordConstExpr          /* 11: ConnectionSpeed */
3107         ',' ClockPolarityKeyword    /* 13: ClockPolarity */
3108         ',' ClockPhaseKeyword       /* 15: ClockPhase */
3109         ',' StringData              /* 17: ResourceSource */
3110         OptionalByteConstExpr       /* 18: ResourceSourceIndex */
3111         OptionalResourceType        /* 19: ResourceType */
3112         OptionalNameString          /* 20: DescriptorName */
3113         OptionalBuffer_Last         /* 21: VendorData */
3114         ')'                         {$$ = TrLinkChildren ($<n>3,13,$4,$5,$6,$8,$
3115     | PARSEOP_SPI_SERIALBUS '('
3116         error ')'                   {$$ = AslDoError(); yyclearin;}
3117     ;

3119 StartDependentFnNoPriTerm
3120     : PARSEOP_STARTDEPENDENTFN_NOPRI '('    {$<n>$ = TrCreateLeafNode (PARSEOP_S
3121         ')' '{'
3122         ResourceMacroList '}'       {$$ = TrLinkChildren ($<n>3,1,$6);}
3123     | PARSEOP_STARTDEPENDENTFN_NOPRI '('
3124         error ')'                   {$$ = AslDoError(); yyclearin;}
3125     ;

3127 StartDependentFnTerm
3128     : PARSEOP_STARTDEPENDENTFN '('  {$<n>$ = TrCreateLeafNode (PARSEOP_STARTDEPE
3129         ByteConstExpr
3130         ',' ByteConstExpr
3131         ')' '{'
3132         ResourceMacroList '}'       {$$ = TrLinkChildren ($<n>3,3,$4,$6,$9);}
3133     | PARSEOP_STARTDEPENDENTFN '('
3134         error ')'                   {$$ = AslDoError(); yyclearin;}
3135     ;

3137 UartSerialBusTerm
3138     : PARSEOP_UART_SERIALBUS '('    {$<n>$ = TrCreateLeafNode (PARSEOP_UART_SERI
3139         DWordConstExpr              /* 04: ConnectionSpeed */
3140         OptionalBitsPerByte         /* 05: BitsPerByte */
3141         OptionalStopBits            /* 06: StopBits */
3142         ',' ByteConstExpr           /* 08: LinesInUse */
3143         OptionalEndian              /* 09: Endianess */
3144         OptionalParityType          /* 10: Parity */
3145         OptionalFlowControl         /* 11: FlowControl */
3146         ',' WordConstExpr           /* 13: Rx BufferSize */
3147         ',' WordConstExpr           /* 15: Tx BufferSize */
3148         ',' StringData              /* 17: ResourceSource */
3149         OptionalByteConstExpr       /* 18: ResourceSourceIndex */
3150         OptionalResourceType        /* 19: ResourceType */
3151         OptionalNameString          /* 20: DescriptorName */
3152         OptionalBuffer_Last         /* 21: VendorData */
3153         ')'                         {$$ = TrLinkChildren ($<n>3,14,$4,$5,$6,$8,$
3154     | PARSEOP_UART_SERIALBUS '('
3155         error ')'                   {$$ = AslDoError(); yyclearin;}
3156     ;

3158 VendorLongTerm
3159     : PARSEOP_VENDORLONG '('        {$<n>$ = TrCreateLeafNode (PARSEOP_VENDORLON
3160         OptionalNameString_First
3161         ')' '{'
3162             ByteList '}'            {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
```

```
3163     | PARSEOP_VENDORLONG '('
3164         error ')'                  {$$ = AslDoError(); yyclearin;}
3165     ;

3167 VendorShortTerm
3168     : PARSEOP_VENDORSHORT '('      {$<n>$ = TrCreateLeafNode (PARSEOP_VENDORSHO
3169         OptionalNameString_First
3170         ')' '{'
3171             ByteList '}'           {$$ = TrLinkChildren ($<n>3,2,$4,$7);}
3172     | PARSEOP_VENDORSHORT '('
3173         error ')'                  {$$ = AslDoError(); yyclearin;}
3174     ;

3176 WordBusNumberTerm
3177     : PARSEOP_WORDBUSNUMBER '('    {$<n>$ = TrCreateLeafNode (PARSEOP_WORDBUSNU
3178         OptionalResourceType_First
3179         OptionalMinType
3180         OptionalMaxType
3181         OptionalDecodeType
3182         ',' WordConstExpr
3183         ',' WordConstExpr
3184         ',' WordConstExpr
3185         ',' WordConstExpr
3186         ',' WordConstExpr
3187         OptionalByteConstExpr
3188         OptionalStringData
3189         OptionalNameString_Last
3190         ')'                        {$$ = TrLinkChildren ($<n>3,12,$4,$5,$6,$7,$
3191     | PARSEOP_WORDBUSNUMBER '('
3192         error ')'                  {$$ = AslDoError(); yyclearin;}
3193     ;

3195 WordIOTerm
3196     : PARSEOP_WORDIO '('           {$<n>$ = TrCreateLeafNode (PARSEOP_WORDIO);}
3197         OptionalResourceType_First
3198         OptionalMinType
3199         OptionalMaxType
3200         OptionalDecodeType
3201         OptionalRangeType
3202         ',' WordConstExpr
3203         ',' WordConstExpr
3204         ',' WordConstExpr
3205         ',' WordConstExpr
3206         ',' WordConstExpr
3207         OptionalByteConstExpr
3208         OptionalStringData
3209         OptionalNameString
3210         OptionalType
3211         OptionalTranslationType_Last
3212         ')'                        {$$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
3213     | PARSEOP_WORDIO '('
3214         error ')'                  {$$ = AslDoError(); yyclearin;}
3215     ;

3217 WordSpaceTerm
3218     : PARSEOP_WORDSPACE '('        {$<n>$ = TrCreateLeafNode (PARSEOP_WORDSPACE
3219         ByteConstExpr              {UtCheckIntegerRange ($4, 0xC0, 0xFF);}
3220         OptionalResourceType
3221         OptionalDecodeType
3222         OptionalMinType
3223         OptionalMaxType
3224         ',' ByteConstExpr
3225         ',' WordConstExpr
3226         ',' WordConstExpr
3227         ',' WordConstExpr
3228         ',' WordConstExpr
```



```
3229         ',' WordConstExpr
3230         OptionalByteConstExpr
3231         OptionalStringData
3232         OptionalNameString_Last
3233         ')'                           {$$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
3234     | PARSEOP_WORDSPACE '('
3235         error ')'                     {$$ = AslDoError(); yyclearin;}
3236     ;


3239 /******* Object References ********************************************/

3241 /* Allow IO, DMA, IRQ Resource macro names to also be used as identifiers */

3243 NameString
3244     : NameSeg                         {}
3245     | PARSEOP_NAMESTRING              {$$ = TrCreateValuedLeafNode (PARSEOP_NAMEST
3246     | PARSEOP_IO                      {$$ = TrCreateValuedLeafNode (PARSEOP_NAMEST
3247     | PARSEOP_DMA                     {$$ = TrCreateValuedLeafNode (PARSEOP_NAMEST
3248     | PARSEOP_IRQ                     {$$ = TrCreateValuedLeafNode (PARSEOP_NAMEST
3249     ;

3251 NameSeg
3252     : PARSEOP_NAMESEG                 {$$ = TrCreateValuedLeafNode (PARSEOP_NAMESE
3253     ;


3256 /******* Helper rules *************************************************/


3259 AmlPackageLengthTerm
3260     : Integer                         {$$ = TrUpdateNode (PARSEOP_PACKAGE_LENGTH,(
3261     ;

3263 NameStringItem
3264     : ',' NameString                  {$$ = $2;}
3265     | ',' error                       {$$ = AslDoError (); yyclearin;}
3266     ;

3268 TermArgItem
3269     : ',' TermArg                     {$$ = $2;}
3270     | ',' error                       {$$ = AslDoError (); yyclearin;}
3271     ;

3273 OptionalBusMasterKeyword
3274     : ','                                       {$$ = TrCreateLeafNode (PARSEOP_
3275     | ',' PARSEOP_BUSMASTERTYPE_MASTER          {$$ = TrCreateLeafNode (PARSEOP_
3276     | ',' PARSEOP_BUSMASTERTYPE_NOTMASTER       {$$ = TrCreateLeafNode (PARSEOP_
3277     ;

3279 OptionalAccessAttribTerm
3280     :                                 {$$ = NULL;}
3281     | ','                             {$$ = NULL;}
3282     | ',' ByteConstExpr               {$$ = $2;}
3283     | ',' AccessAttribKeyword         {$$ = $2;}
3284     ;

3286 OptionalAccessSize
3287     :                                 {$$ = TrCreateValuedLeafNode (PARSEOP_BYTECO
3288     | ','                             {$$ = TrCreateValuedLeafNode (PARSEOP_BYTECO
3289     | ',' ByteConstExpr               {$$ = $2;}
3290     ;

3292 OptionalAddressingMode
3293     : ','                             {$$ = NULL;}
3294     | ',' AddressingModeKeyword       {$$ = $2;}
```

```
3295     ;

3297 OptionalAddressRange
3298     :                               {$$ = NULL;}
3299     | ',' 
3300     | ',' AddressKeyword            {$$ = $2;}
3301     ;

3303 OptionalBitsPerByte
3304     : ','                           {$$ = NULL;}
3305     | ',' BitsPerByteKeyword        {$$ = $2;}
3306     ;

3308 OptionalBuffer_Last
3309     :                               {$$ = NULL;}
3310     | ','                           {$$ = NULL;}
3311     | ',' DataBufferTerm            {$$ = $2;}
3312     ;

3314 OptionalByteConstExpr
3315     :                               {$$ = NULL;}
3316     | ','                           {$$ = NULL;}
3317     | ',' ByteConstExpr             {$$ = $2;}
3318     ;

3320 OptionalDecodeType
3321     : ','                           {$$ = NULL;}
3322     | ',' DecodeKeyword             {$$ = $2;}
3323     ;

3325 OptionalDevicePolarity
3326     : ','                           {$$ = NULL;}
3327     | ',' DevicePolarityKeyword     {$$ = $2;}
3328     ;

3330 OptionalDWordConstExpr
3331     :                               {$$ = NULL;}
3332     | ','                           {$$ = NULL;}
3333     | ',' DWordConstExpr            {$$ = $2;}
3334     ;

3336 OptionalEndian
3337     : ','                           {$$ = NULL;}
3338     | ',' EndianKeyword             {$$ = $2;}
3339     ;

3341 OptionalFlowControl
3342     : ','                           {$$ = NULL;}
3343     | ',' FlowControlKeyword        {$$ = $2;}
3344     ;

3346 OptionalIoRestriction
3347     : ','                           {$$ = NULL;}
3348     | ',' IoRestrictionKeyword      {$$ = $2;}
3349     ;

3351 OptionalListString
3352     :                               {$$ = TrCreateValuedLeafNode (PARSEOP_STRING
3353     | ','                           {$$ = TrCreateValuedLeafNode (PARSEOP_STRING
3354     | ',' TermArg                   {$$ = $2;}
3355     ;

3357 OptionalMaxType
3358     : ','                           {$$ = NULL;}
3359     | ',' MaxKeyword                {$$ = $2;}
3360     ;
```



```
3362 OptionalMemType
3363     : ','                           {$$ = NULL;}
3364     | ',' MemTypeKeyword            {$$ = $2;}
3365     ;

3367 OptionalMinType
3368     : ','                           {$$ = NULL;}
3369     | ',' MinKeyword                {$$ = $2;}
3370     ;

3372 OptionalNameString
3373     :                               {$$ = NULL;}
3374     | ','                           {$$ = NULL;}
3375     | ',' NameString                {$$ = $2;}
3376     ;

3378 OptionalNameString_Last
3379     :                               {$$ = NULL;}
3380     | ','                           {$$ = NULL;}
3381     | ',' NameString                {$$ = $2;}
3382     ;

3384 OptionalNameString_First
3385     :                               {$$ = TrCreateLeafNode (PARSEOP_ZERO);}
3386     | NameString                    {$$ = $1;}
3387     ;

3389 OptionalObjectTypeKeyword
3390     :                               {$$ = TrCreateLeafNode (PARSEOP_OBJECTTYPE_U
3391     | ',' ObjectTypeKeyword         {$$ = $2;}
3392     ;

3394 OptionalParityType
3395     : ','                           {$$ = NULL;}
3396     | ',' ParityTypeKeyword         {$$ = $2;}
3397     ;

3399 OptionalQWordConstExpr
3400     :                               {$$ = NULL;}
3401     | ','                           {$$ = NULL;}
3402     | ',' QWordConstExpr            {$$ = $2;}
3403     ;

3405 OptionalRangeType
3406     : ','                           {$$ = NULL;}
3407     | ',' RangeTypeKeyword          {$$ = $2;}
3408     ;

3410 OptionalReadWriteKeyword
3411     :                                   {$$ = TrCreateLeafNode (PARSEOP_READWRIT
3412     | PARSEOP_READWRITETYPE_BOTH        {$$ = TrCreateLeafNode (PARSEOP_READWRIT
3413     | PARSEOP_READWRITETYPE_READONLY    {$$ = TrCreateLeafNode (PARSEOP_READWRIT
3414     ;

3416 OptionalReference
3417     :                               {$$ = TrCreateLeafNode (PARSEOP_ZERO);}
3418     | ','                           {$$ = TrCreateLeafNode (PARSEOP_ZERO);}
3419     | ',' TermArg                   {$$ = $2;}
3420     ;

3422 OptionalResourceType_First
3423     :                               {$$ = TrCreateLeafNode (PARSEOP_RESOURCETYPE
3424     | ResourceTypeKeyword           {$$ = $1;}
3425     ;
```

```
3427 OptionalResourceType
3428     :                              {$$ = TrCreateLeafNode (PARSEOP_RESOURCETYPE
3429     | ',' ,                        {$$ = TrCreateLeafNode (PARSEOP_RESOURCETYPE
3430     | ',' ResourceTypeKeyword      {$$ = $2;}
3431     ;

3433 OptionalReturnArg
3434     :                              {$$ = TrSetNodeFlags (TrCreateLeafNode (PARS
3435     | TermArg                      {$$ = $1;}
3436     ;

3438 OptionalSerializeRuleKeyword
3439     :                              {$$ = NULL;}
3440     | ',' ,                        {$$ = NULL;}
3441     | ',' SerializeRuleKeyword     {$$ = $2;}
3442     ;

3444 OptionalSlaveMode
3445     : ',' ,                        {$$ = NULL;}
3446     | ',' SlaveModeKeyword         {$$ = $2;}
3447     ;

3449 OptionalShareType
3450     :                              {$$ = NULL;}
3451     | ',' ,                        {$$ = NULL;}
3452     | ',' ShareTypeKeyword         {$$ = $2;}
3453     ;

3455 OptionalShareType_First
3456     :                              {$$ = NULL;}
3457     | ShareTypeKeyword             {$$ = $1;}
3458     ;

3460 OptionalStopBits
3461     : ',' ,                        {$$ = NULL;}
3462     | ',' StopBitsKeyword          {$$ = $2;}
3463     ;

3465 OptionalStringData
3466     :                              {$$ = NULL;}
3467     | ',' ,                        {$$ = NULL;}
3468     | ',' StringData               {$$ = $2;}
3469     ;

3471 OptionalTermArg
3472     :                              {$$ = NULL;}
3473     | TermArg                      {$$ = $1;}
3474     ;

3476 OptionalType
3477     :                              {$$ = NULL;}
3478     | ',' ,                        {$$ = NULL;}
3479     | ',' TypeKeyword              {$$ = $2;}
3480     ;

3482 OptionalType_Last
3483     :                              {$$ = NULL;}
3484     | ',' ,                        {$$ = NULL;}
3485     | ',' TypeKeyword              {$$ = $2;}
3486     ;

3488 OptionalTranslationType_Last
3489     :                              {$$ = NULL;}
3490     | ',' ,                        {$$ = NULL;}
3491     | ',' TranslationKeyword       {$$ = $2;}
3492     ;
```



```
3494 OptionalWireMode
3495     : ',' ,                        {$$ = NULL;}
3496     | ',' WireModeKeyword          {$$ = $2;}
3497     ;

3499 OptionalWordConst
3500     :                              {$$ = NULL;}
3501     | WordConst                    {$$ = $1;}
3502     ;

3504 OptionalWordConstExpr
3505     : ',' ,                        {$$ = NULL;}
3506     | ',' WordConstExpr            {$$ = $2;}
3507     ;

3509 OptionalXferSize
3510     :                              {$$ = TrCreateValuedLeafNode (PARSEOP_XFERSI
3511     | ',' ,                        {$$ = TrCreateValuedLeafNode (PARSEOP_XFERSI
3512     | ',' XferSizeKeyword          {$$ = $2;}
3513     ;

3515 %%
3516 /*****************************************************************************
3517  *
3518  * Local support functions
3519  *
3520  ****************************************************************************/

3522 int
3523 AslCompilerwrap(void)
3524 {
3525   return (1);
3526 }

3528 /*! [End] no source code translation !*/

3530 void *
3531 AslLocalAllocate (unsigned int Size)
3532 {
3533     void                *Mem;


3536     DbgPrint (ASL_PARSE_OUTPUT, "\nAslLocalAllocate: Expanding Stack to %u\n\n",

3538     Mem = ACPI_ALLOCATE_ZEROED (Size);
3539     if (!Mem)
3540     {
3541         AslCommonError (ASL_ERROR, ASL_MSG_MEMORY_ALLOCATION,
3542                         Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
3543                         Gbl_InputByteCount, Gbl_CurrentColumn,
3544                         Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
3545         exit (1);
3546     }

3548     return (Mem);
3549 }

3551 ACPI_PARSE_OBJECT *
3552 AslDoError (void)
3553 {


3556     return (TrCreateLeafNode (PARSEOP_ERRORNODE));

3558 }
```

```
3561 /*******************************************************************************
3562  *
3563  * FUNCTION:    UtGetOpName
3564  *
3565  * PARAMETERS:  ParseOpcode         - Parser keyword ID
3566  *
3567  * RETURN:      Pointer to the opcode name
3568  *
3569  * DESCRIPTION: Get the ascii name of the parse opcode
3570  *
3571  ******************************************************************************/

3573 char *
3574 UtGetOpName (
3575     UINT32                  ParseOpcode)
3576 {
3577 #ifdef ASL_YYTNAME_START
3578     /*
3579      * First entries (ASL_YYTNAME_START) in yytname are special reserved names.
3580      * Ignore first 8 characters of the name
3581      */
3582     return ((char *) yytname
3583         [(ParseOpcode - ASL_FIRST_PARSE_OPCODE) + ASL_YYTNAME_START] + 8);
3584 #else
3585     return ("[Unknown parser generator]");
3586 #endif
3587 }
```

```
**********************************************************
    6470 Thu Dec 26 13:48:27 2013
new/usr/src/common/acpica/compiler/asldefine.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asldefine.h - Common defines for the iASL compiler
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ASLDEFINE_H
#define __ASLDEFINE_H


/*
 * Compiler versions and names
 */
#define ASL_REVISION                ACPI_CA_VERSION
#define ASL_COMPILER_NAME           "ASL Optimizing Compiler"
#define AML_DISASSEMBLER_NAME       "AML Disassembler"
#define ASL_INVOCATION_NAME         "iasl"
#define ASL_CREATOR_ID              "INTL"

#define ASL_COMPLIANCE              "Supports ACPI Specification Revision 5.0A"
```



```
/* Configuration constants */

#define ASL_MAX_ERROR_COUNT         200
#define ASL_NODE_CACHE_SIZE         1024
#define ASL_STRING_CACHE_SIZE       32768

#define ASL_FIRST_PARSE_OPCODE      PARSEOP_ACCESSAS
#define ASL_PARSE_OPCODE_BASE       PARSEOP_ACCESSAS        /* First Lex type */


/*
 * Per-parser-generator configuration. These values are used to cheat and
 * directly access the bison/yacc token name table (yyname or yytname).
 * Note: These values are the index in yyname for the first lex token
 * (PARSEOP_ACCCESSAS).
 */
#if defined (YYBISON)
#define ASL_YYTNAME_START           3   /* Bison */
#elif defined (YYBYACC)
#define ASL_YYTNAME_START           257 /* Berkeley yacc */
#endif


/*
 * Macros
 */
#define ASL_RESDESC_OFFSET(m)       ACPI_OFFSET (AML_RESOURCE, m)
#define ASL_PTR_DIFF(a,b)           ((UINT8 *)(b) - (UINT8 *)(a))
#define ASL_PTR_ADD(a,b)            ((UINT8 *)(a) = ((UINT8 *)(a) + (b)))
#define ASL_GET_CHILD_NODE(a)       (a)->Asl.Child
#define ASL_GET_PEER_NODE(a)        (a)->Asl.Next
#define OP_TABLE_ENTRY(a,b,c,d)     {b,d,a,c}


/* Internal AML opcodes */

#define AML_RAW_DATA_BYTE           (UINT16) 0xAA01 /* write one raw byte */
#define AML_RAW_DATA_WORD           (UINT16) 0xAA02 /* write 2 raw bytes */
#define AML_RAW_DATA_DWORD          (UINT16) 0xAA04 /* write 4 raw bytes */
#define AML_RAW_DATA_QWORD          (UINT16) 0xAA08 /* write 8 raw bytes */
#define AML_RAW_DATA_BUFFER         (UINT16) 0xAA0B /* raw buffer with length */
#define AML_RAW_DATA_CHAIN          (UINT16) 0xAA0C /* chain of raw buffers */
#define AML_PACKAGE_LENGTH          (UINT16) 0xAA10
#define AML_UNASSIGNED_OPCODE       (UINT16) 0xEEEE
#define AML_DEFAULT_ARG_OP          (UINT16) 0xDDDD


/* filename suffixes for output files */

#define FILE_SUFFIX_PREPROCESSOR    "i"
#define FILE_SUFFIX_AML_CODE        "aml"
#define FILE_SUFFIX_LISTING         "lst"
#define FILE_SUFFIX_HEX_DUMP        "hex"
#define FILE_SUFFIX_DEBUG           "txt"
#define FILE_SUFFIX_SOURCE          "src"
#define FILE_SUFFIX_NAMESPACE       "nsp"
#define FILE_SUFFIX_ASM_SOURCE      "asm"
#define FILE_SUFFIX_C_SOURCE        "c"
#define FILE_SUFFIX_DISASSEMBLY     "dsl"
#define FILE_SUFFIX_ASM_INCLUDE     "inc"
#define FILE_SUFFIX_C_INCLUDE       "h"
#define FILE_SUFFIX_ASL_CODE        "asl"
#define FILE_SUFFIX_C_OFFSET        "offset.h"
```

```
126 /* Types for input files */

128 #define ASL_INPUT_TYPE_BINARY       0
129 #define ASL_INPUT_TYPE_ACPI_TABLE   1
130 #define ASL_INPUT_TYPE_ASCII_ASL    2
131 #define ASL_INPUT_TYPE_ASCII_DATA   3


134 /* Misc */

136 #define ASL_EXTERNAL_METHOD         255
137 #define ASL_ABORT                   TRUE
138 #define ASL_NO_ABORT                FALSE
139 #define ASL_EOF                     ACPI_UINT32_MAX


142 /* Listings */

144 #define ASL_LISTING_LINE_PREFIX         ":  "


147 /* Support for reserved method names */

149 #define ACPI_VALID_RESERVED_NAME_MAX    0x80000000
150 #define ACPI_NOT_RESERVED_NAME          ACPI_UINT32_MAX
151 #define ACPI_PREDEFINED_NAME            (ACPI_UINT32_MAX - 1)
152 #define ACPI_EVENT_RESERVED_NAME        (ACPI_UINT32_MAX - 2)
153 #define ACPI_COMPILER_RESERVED_NAME     (ACPI_UINT32_MAX - 3)


156 /* String to Integer conversion */

158 #define NEGATIVE                    1
159 #define POSITIVE                    0


162 /* Helper macros for resource tag creation */

164 #define RsCreateMultiBitField \
165     RsCreateResourceField

167 #define RsCreateBitField(Op, Name, ByteOffset, BitOffset) \
168     RsCreateResourceField (Op, Name, ByteOffset, BitOffset, 1)

170 #define RsCreateByteField(Op, Name, ByteOffset) \
171     RsCreateResourceField (Op, Name, ByteOffset, 0, 8);

173 #define RsCreateWordField(Op, Name, ByteOffset) \
174     RsCreateResourceField (Op, Name, ByteOffset, 0, 16);

176 #define RsCreateDwordField(Op, Name, ByteOffset) \
177     RsCreateResourceField (Op, Name, ByteOffset, 0, 32);

179 #define RsCreateQwordField(Op, Name, ByteOffset) \
180     RsCreateResourceField (Op, Name, ByteOffset, 0, 64);

182 #endif /* ASLDEFINE.H */
```

```
**********************************************************
   26638 Thu Dec 26 13:48:27 2013
new/usr/src/common/acpica/compiler/aslerror.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: aslerror - Error handling and statistics
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define ASL_EXCEPTIONS
  45 #include "aslcompiler.h"

  47 #define _COMPONENT          ACPI_COMPILER
  48         ACPI_MODULE_NAME    ("aslerror")

  50 /* Local prototypes */

  52 static void
  53 AeAddToErrorLog (
  54     ASL_ERROR_MSG           *Enode);


  57 /******************************************************************************
  58  *
```



```
  59  * FUNCTION:    AeClearErrorLog
  60  *
  61  * PARAMETERS:  None
  62  *
  63  * RETURN:      None
  64  *
  65  * DESCRIPTION: Empty the error list
  66  *
  67  *****************************************************************************/

  69 void
  70 AeClearErrorLog (
  71     void)
  72 {
  73     ASL_ERROR_MSG           *Enode = Gbl_ErrorLog;
  74     ASL_ERROR_MSG           *Next;

  76     /* Walk the error node list */

  78     while (Enode)
  79     {
  80         Next = Enode->Next;
  81         ACPI_FREE (Enode);
  82         Enode = Next;
  83     }

  85     Gbl_ErrorLog = NULL;
  86 }


  89 /******************************************************************************
  90  *
  91  * FUNCTION:    AeAddToErrorLog
  92  *
  93  * PARAMETERS:  Enode       - An error node to add to the log
  94  *
  95  * RETURN:      None
  96  *
  97  * DESCRIPTION: Add a new error node to the error log. The error log is
  98  *              ordered by the "logical" line number (cumulative line number
  99  *              including all include files.)
 100  *
 101  *****************************************************************************/

 103 static void
 104 AeAddToErrorLog (
 105     ASL_ERROR_MSG           *Enode)
 106 {
 107     ASL_ERROR_MSG           *Next;
 108     ASL_ERROR_MSG           *Prev;


 111     /* If Gbl_ErrorLog is null, this is the first error node */

 113     if (!Gbl_ErrorLog)
 114     {
 115         Gbl_ErrorLog = Enode;
 116         return;
 117     }

 119     /*
 120      * Walk error list until we find a line number greater than ours.
 121      * List is sorted according to line number.
 122      */
 123     Prev = NULL;
 124     Next = Gbl_ErrorLog;
```

```
126     while ((Next) &&
127            (Next->LogicalLineNumber <= Enode->LogicalLineNumber))
128     {
129         Prev = Next;
130         Next = Next->Next;
131     }

133     /* Found our place in the list */

135     Enode->Next = Next;

137     if (Prev)
138     {
139         Prev->Next = Enode;
140     }
141     else
142     {
143         Gbl_ErrorLog = Enode;
144     }
145 }


148 /******************************************************************************
149  *
150  * FUNCTION:    AePrintException
151  *
152  * PARAMETERS:  FileId          - ID of output file
153  *              Enode           - Error node to print
154  *              Header          - Additional text before each message
155  *
156  * RETURN:      None
157  *
158  * DESCRIPTION: Print the contents of an error node.
159  *
160  * NOTE:        We don't use the FlxxxFile I/O functions here because on error
161  *              they abort the compiler and call this function!  Since we
162  *              are reporting errors here, we ignore most output errors and
163  *              just try to get out as much as we can.
164  *
165  *****************************************************************************/

167 void
168 AePrintException (
169     UINT32                  FileId,
170     ASL_ERROR_MSG           *Enode,
171     char                    *Header)
172 {
173     UINT8                   SourceByte;
174     int                     Actual;
175     size_t                  RActual;
176     UINT32                  MsgLength;
177     char                    *MainMessage;
178     char                    *ExtraMessage;
179     UINT32                  SourceColumn;
180     UINT32                  ErrorColumn;
181     FILE                    *OutputFile;
182     FILE                    *SourceFile = NULL;
183     long                    FileSize;
184     BOOLEAN                 PrematureEOF = FALSE;
185     UINT32                  Total = 0;


188     if (Gbl_NoErrors)
189     {
190         return;
```



```
191     }

193     /*
194      * Only listing files have a header, and remarks/optimizations
195      * are always output
196      */
197     if (!Header)
198     {
199         /* Ignore remarks if requested */

201         switch (Enode->Level)
202         {
203         case ASL_WARNING:
204         case ASL_WARNING2:
205         case ASL_WARNING3:

207             if (!Gbl_DisplayWarnings)
208             {
209                 return;
210             }
211             break;

213         case ASL_REMARK:

215             if (!Gbl_DisplayRemarks)
216             {
217                 return;
218             }
219             break;

221         case ASL_OPTIMIZATION:

223             if (!Gbl_DisplayOptimizations)
224             {
225                 return;
226             }
227             break;

229         default:

231             break;
232         }
233     }

235     /* Get the various required file handles */

237     OutputFile = Gbl_Files[FileId].Handle;

239     if (!Enode->SourceLine)
240     {
241         /* Use the merged header/source file if present, otherwise use input fil

243         SourceFile = Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Handle;
244         if (!SourceFile)
245         {
246             SourceFile = Gbl_Files[ASL_FILE_INPUT].Handle;
247         }

249         if (SourceFile)
250         {
251             /* Determine if the error occurred at source file EOF */

253             fseek (SourceFile, 0, SEEK_END);
254             FileSize = ftell (SourceFile);

256             if ((long) Enode->LogicalByteOffset >= FileSize)
```

```
257             {
258                 PrematureEOF = TRUE;
259             }
260         }
261     }

263     if (Header)
264     {
265         fprintf (OutputFile, "%s", Header);
266     }

268     /* Print filename and line number if present and valid */

270     if (Enode->Filename)
271     {
272         if (Gbl_VerboseErrors)
273         {
274             fprintf (OutputFile, "%-8s", Enode->Filename);

276             if (Enode->LineNumber)
277             {
278                 if (Enode->SourceLine)
279                 {
280                     fprintf (OutputFile, " %6u: %s",
281                         Enode->LineNumber, Enode->SourceLine);
282                 }
283                 else
284                 {
285                     fprintf (OutputFile, " %6u: ", Enode->LineNumber);

287                     /*
288                      * If not at EOF, get the corresponding source code line and
289                      * display it. Don't attempt this if we have a premature EOF
290                      * condition.
291                      */
292                     if (!PrematureEOF)
293                     {
294                         /*
295                          * Seek to the offset in the combined source file, read
296                          * the source line, and write it to the output.
297                          */
298                         Actual = fseek (SourceFile, (long) Enode->LogicalByteOff
299                                     (int) SEEK_SET);
300                         if (Actual)
301                         {
302                             fprintf (OutputFile,
303                                 "[*** iASL: Seek error on source code temp file
304                                 Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Filename);
305                         }
306                         else
307                         {
308                             RActual = fread (&SourceByte, 1, 1, SourceFile);
309                             if (RActual != 1)
310                             {
311                                 fprintf (OutputFile,
312                                     "[*** iASL: Read error on source code temp f
313                                     Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Filename);
314                             }
315                             else
316                             {
317                                 /* Read/write the source line, up to the maximum

319                                 while (RActual && SourceByte && (SourceByte != '
320                                 {
321                                     if (Total < 256)
322                                     {
```



```
323                                         /* After the max line length, we will ju

325                                         if (fwrite (&SourceByte, 1, 1, OutputFil
326                                         {
327                                             printf ("[*** iASL: Write error on o
328                                             return;
329                                         }
330                                     }
331                                     else if (Total == 256)
332                                     {
333                                         fprintf (OutputFile,
334                                             "\n[*** iASL: Very long input line,
335                                             Enode->Column);
336                                     }

338                                     RActual = fread (&SourceByte, 1, 1, SourceFi
339                                     if (RActual != 1)
340                                     {
341                                         fprintf (OutputFile,
342                                             "[*** iASL: Read error on source cod
343                                             Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Fi
344                                         return;
345                                     }
346                                     Total++;
347                                 }
348                             }
349                         }
350                     }

352                     fprintf (OutputFile, "\n");
353                 }
354             }
355         }
356         else
357         {
358             /*
359              * Less verbose version of the error message, enabled via the
360              * -vi switch. The format is compatible with MS Visual Studio.
361              */
362             fprintf (OutputFile, "%s", Enode->Filename);

364             if (Enode->LineNumber)
365             {
366                 fprintf (OutputFile, "(%u) : ",
367                     Enode->LineNumber);
368             }
369         }
370     }

372     /* NULL message ID, just print the raw message */

374     if (Enode->MessageId == 0)
375     {
376         fprintf (OutputFile, "%s\n", Enode->Message);
377     }
378     else
379     {
380         /* Decode the message ID */

382         if (Gbl_VerboseErrors)
383         {
384             fprintf (OutputFile, "%s %4.4d -",
385                         AslErrorLevel[Enode->Level],
386                         Enode->MessageId + ((Enode->Level+1) * 1000));
387         }
388         else /* IDE case */
```

```
389         {
390             fprintf (OutputFile, "%s %4.4d:",
391                         AslErrorLevelIde[Enode->Level],
392                         Enode->MessageId + ((Enode->Level+1) * 1000));
393         }
394
395         MainMessage = AslMessages[Enode->MessageId];
396         ExtraMessage = Enode->Message;
397
398         if (Enode->LineNumber)
399         {
400             /* Main message: try to use string from AslMessages first */
401
402             if (!MainMessage)
403             {
404                 MainMessage = "";
405             }
406
407             MsgLength = strlen (MainMessage);
408             if (MsgLength == 0)
409             {
410                 /* Use the secondary/extra message as main message */
411
412                 MainMessage = Enode->Message;
413                 if (!MainMessage)
414                 {
415                     MainMessage = "";
416                 }
417
418                 MsgLength = strlen (MainMessage);
419                 ExtraMessage = NULL;
420             }
421
422             if (Gbl_VerboseErrors && !PrematureEOF)
423             {
424                 if (Total >= 256)
425                 {
426                     fprintf (OutputFile, "    %s",
427                         MainMessage);
428                 }
429                 else
430                 {
431                     SourceColumn = Enode->Column + Enode->FilenameLength + 6 + 2
432                     ErrorColumn = ASL_ERROR_LEVEL_LENGTH + 5 + 2 + 1;
433
434                     if ((MsgLength + ErrorColumn) < (SourceColumn - 1))
435                     {
436                         fprintf (OutputFile, "%*s%s",
437                             (int) ((SourceColumn - 1) - ErrorColumn),
438                             MainMessage, " ^ ");
439                     }
440                     else
441                     {
442                         fprintf (OutputFile, "%*s %s",
443                             (int) ((SourceColumn - ErrorColumn) + 1), "^",
444                             MainMessage);
445                     }
446                 }
447             }
448             else
449             {
450                 fprintf (OutputFile, " %s", MainMessage);
451             }
452
453             /* Print the extra info message if present */
```



```
455             if (ExtraMessage)
456             {
457                 fprintf (OutputFile, " (%s)", ExtraMessage);
458             }
459
460             if (PrematureEOF)
461             {
462                 fprintf (OutputFile, " and premature End-Of-File");
463             }
464
465             fprintf (OutputFile, "\n");
466             if (Gbl_VerboseErrors)
467             {
468                 fprintf (OutputFile, "\n");
469             }
470         }
471         else
472         {
473             fprintf (OutputFile, " %s %s\n\n", MainMessage, ExtraMessage);
474         }
475     }
476 }


479 /******************************************************************************
480  *
481  * FUNCTION:    AePrintErrorLog
482  *
483  * PARAMETERS:  FileId          - Where to output the error log
484  *
485  * RETURN:      None
486  *
487  * DESCRIPTION: Print the entire contents of the error log
488  *
489  *****************************************************************************/

491 void
492 AePrintErrorLog (
493     UINT32                  FileId)
494 {
495     ASL_ERROR_MSG           *Enode = Gbl_ErrorLog;


498     /* Walk the error node list */

500     while (Enode)
501     {
502         AePrintException (FileId, Enode, NULL);
503         Enode = Enode->Next;
504     }
505 }


508 /******************************************************************************
509  *
510  * FUNCTION:    AslCommonError2
511  *
512  * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
513  *              MessageId           - Index into global message buffer
514  *              LineNumber          - Actual file line number
515  *              Column              - Column in current line
516  *              SourceLine          - Actual source code line
517  *              Filename            - source filename
518  *              ExtraMessage        - additional error message
519  *
520  * RETURN:      None
```

```
 *
 * DESCRIPTION: Create a new error node and add it to the error log
 *
 ******************************************************************************/

void
AslCommonError2 (
    UINT8                   Level,
    UINT8                   MessageId,
    UINT32                  LineNumber,
    UINT32                  Column,
    char                    *SourceLine,
    char                    *Filename,
    char                    *ExtraMessage)
{
    char                    *MessageBuffer = NULL;
    char                    *LineBuffer;
    ASL_ERROR_MSG           *Enode;


    Enode = UtLocalCalloc (sizeof (ASL_ERROR_MSG));

    if (ExtraMessage)
    {
        /* Allocate a buffer for the message and a new error node */

        MessageBuffer = UtLocalCalloc (strlen (ExtraMessage) + 1);

        /* Keep a copy of the extra message */

        ACPI_STRCPY (MessageBuffer, ExtraMessage);
    }

    LineBuffer = UtLocalCalloc (strlen (SourceLine) + 1);
    ACPI_STRCPY (LineBuffer, SourceLine);

    /* Initialize the error node */

    if (Filename)
    {
        Enode->Filename       = Filename;
        Enode->FilenameLength = strlen (Filename);
        if (Enode->FilenameLength < 6)
        {
            Enode->FilenameLength = 6;
        }
    }

    Enode->MessageId            = MessageId;
    Enode->Level                = Level;
    Enode->LineNumber           = LineNumber;
    Enode->LogicalLineNumber    = LineNumber;
    Enode->LogicalByteOffset    = 0;
    Enode->Column               = Column;
    Enode->Message              = MessageBuffer;
    Enode->SourceLine           = LineBuffer;

    /* Add the new node to the error node list */

    AeAddToErrorLog (Enode);

    if (Gbl_DebugFlag)
    {
        /* stderr is a file, send error to it immediately */

        AePrintException (ASL_FILE_STDERR, Enode, NULL);
```



```
    }

    Gbl_ExceptionCount[Level]++;
}


/******************************************************************************
 *
 * FUNCTION:    AslCommonError
 *
 * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
 *              MessageId           - Index into global message buffer
 *              CurrentLineNumber   - Actual file line number
 *              LogicalLineNumber   - Cumulative line number
 *              LogicalByteOffset   - Byte offset in source file
 *              Column              - Column in current line
 *              Filename            - source filename
 *              ExtraMessage        - additional error message
 *
 * RETURN:      None
 *
 * DESCRIPTION: Create a new error node and add it to the error log
 *
 ******************************************************************************/

void
AslCommonError (
    UINT8                   Level,
    UINT8                   MessageId,
    UINT32                  CurrentLineNumber,
    UINT32                  LogicalLineNumber,
    UINT32                  LogicalByteOffset,
    UINT32                  Column,
    char                    *Filename,
    char                    *ExtraMessage)
{
    char                    *MessageBuffer = NULL;
    ASL_ERROR_MSG           *Enode;


    Enode = UtLocalCalloc (sizeof (ASL_ERROR_MSG));

    if (ExtraMessage)
    {
        /* Allocate a buffer for the message and a new error node */

        MessageBuffer = UtLocalCalloc (strlen (ExtraMessage) + 1);

        /* Keep a copy of the extra message */

        ACPI_STRCPY (MessageBuffer, ExtraMessage);
    }

    /* Initialize the error node */

    if (Filename)
    {
        Enode->Filename       = Filename;
        Enode->FilenameLength = strlen (Filename);
        if (Enode->FilenameLength < 6)
        {
            Enode->FilenameLength = 6;
        }
    }

    Enode->MessageId            = MessageId;
```

```
653     Enode->Level                = Level;
654     Enode->LineNumber           = CurrentLineNumber;
655     Enode->LogicalLineNumber    = LogicalLineNumber;
656     Enode->LogicalByteOffset    = LogicalByteOffset;
657     Enode->Column               = Column;
658     Enode->Message              = MessageBuffer;
659     Enode->SourceLine           = NULL;

661     /* Add the new node to the error node list */

663     AeAddToErrorLog (Enode);

665     if (Gbl_DebugFlag)
666     {
667         /* stderr is a file, send error to it immediately */

669         AePrintException (ASL_FILE_STDERR, Enode, NULL);
670     }

672     Gbl_ExceptionCount[Level]++;
673     if (Gbl_ExceptionCount[ASL_ERROR] > ASL_MAX_ERROR_COUNT)
674     {
675         printf ("\nMaximum error count (%u) exceeded\n", ASL_MAX_ERROR_COUNT);

677         Gbl_SourceLine = 0;
678         Gbl_NextError = Gbl_ErrorLog;
679         CmCleanupAndExit ();
680         exit(1);
681     }

683     return;
684 }


687 /******************************************************************************
688  *
689  * FUNCTION:    AslDisableException
690  *
691  * PARAMETERS:  MessageIdString     - ID to be disabled
692  *
693  * RETURN:      Status
694  *
695  * DESCRIPTION: Enter a message ID into the global disabled messages table
696  *
697  *****************************************************************************/

699 ACPI_STATUS
700 AslDisableException (
701     char                    *MessageIdString)
702 {
703     UINT32                  MessageId;


706     /* Convert argument to an integer and validate it */

708     MessageId = (UINT32) strtoul (MessageIdString, NULL, 0);

710     if ((MessageId < 2000) || (MessageId > 5999))
711     {
712         printf ("\"%s\" is not a valid warning/remark ID\n",
713             MessageIdString);
714         return (AE_BAD_PARAMETER);
715     }

717     /* Insert value into the global disabled message array */
```



```
719     if (Gbl_DisabledMessagesIndex >= ASL_MAX_DISABLED_MESSAGES)
720     {
721         printf ("Too many messages have been disabled (max %u)\n",
722             ASL_MAX_DISABLED_MESSAGES);
723         return (AE_LIMIT);
724     }

726     Gbl_DisabledMessages[Gbl_DisabledMessagesIndex] = MessageId;
727     Gbl_DisabledMessagesIndex++;
728     return (AE_OK);
729 }


732 /******************************************************************************
733  *
734  * FUNCTION:    AslIsExceptionDisabled
735  *
736  * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
737  *              MessageId           - Index into global message buffer
738  *
739  * RETURN:      TRUE if exception/message should be ignored
740  *
741  * DESCRIPTION: Check if the user has specified options such that this
742  *              exception should be ignored
743  *
744  *****************************************************************************/

746 BOOLEAN
747 AslIsExceptionDisabled (
748     UINT8                   Level,
749     UINT8                   MessageId)
750 {
751     UINT32                  EncodedMessageId;
752     UINT32                  i;


755     switch (Level)
756     {
757     case ASL_WARNING2:
758     case ASL_WARNING3:

760         /* Check for global disable via -w1/-w2/-w3 options */

762         if (Level > Gbl_WarningLevel)
763         {
764             return (TRUE);
765         }
766         /* Fall through */

768     case ASL_WARNING:
769     case ASL_REMARK:
770         /*
771          * Ignore this warning/remark if it has been disabled by
772          * the user (-vw option)
773          */
774         EncodedMessageId = MessageId + ((Level + 1) * 1000);
775         for (i = 0; i < Gbl_DisabledMessagesIndex; i++)
776         {
777             /* Simple implementation via fixed array */

779             if (EncodedMessageId == Gbl_DisabledMessages[i])
780             {
781                 return (TRUE);
782             }
783         }
784         break;
```

```
786     default:
787         break;
788     }

790     return (FALSE);
791 }


794 /*******************************************************************************
795  *
796  * FUNCTION:    AslError
797  *
798  * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
799  *              MessageId           - Index into global message buffer
800  *              Op                  - Parse node where error happened
801  *              ExtraMessage        - additional error message
802  *
803  * RETURN:      None
804  *
805  * DESCRIPTION: Main error reporting routine for the ASL compiler (all code
806  *              except the parser.)
807  *
808  ******************************************************************************/

810 void
811 AslError (
812     UINT8                   Level,
813     UINT8                   MessageId,
814     ACPI_PARSE_OBJECT       *Op,
815     char                    *ExtraMessage)
816 {

818     /* Check if user wants to ignore this exception */

820     if (AslIsExceptionDisabled (Level, MessageId))
821     {
822         return;
823     }

825     if (Op)
826     {
827         AslCommonError (Level, MessageId, Op->Asl.LineNumber,
828             Op->Asl.LogicalLineNumber,
829             Op->Asl.LogicalByteOffset,
830             Op->Asl.Column,
831             Op->Asl.Filename, ExtraMessage);
832     }
833     else
834     {
835         AslCommonError (Level, MessageId, 0,
836             0, 0, 0, NULL, ExtraMessage);
837     }
838 }


841 /*******************************************************************************
842  *
843  * FUNCTION:    AslCoreSubsystemError
844  *
845  * PARAMETERS:  Op                  - Parse node where error happened
846  *              Status              - The ACPI CA Exception
847  *              ExtraMessage        - additional error message
848  *              Abort               - TRUE -> Abort compilation
849  *
850  * RETURN:      None
```



```
851  *
852  * DESCRIPTION: Error reporting routine for exceptions returned by the ACPI
853  *              CA core subsystem.
854  *
855  ******************************************************************************/

857 void
858 AslCoreSubsystemError (
859     ACPI_PARSE_OBJECT       *Op,
860     ACPI_STATUS             Status,
861     char                    *ExtraMessage,
862     BOOLEAN                 Abort)
863 {

865     sprintf (MsgBuffer, "%s %s", AcpiFormatException (Status), ExtraMessage);

867     if (Op)
868     {
869         AslCommonError (ASL_ERROR, ASL_MSG_CORE_EXCEPTION, Op->Asl.LineNumber,
870                         Op->Asl.LogicalLineNumber,
871                         Op->Asl.LogicalByteOffset,
872                         Op->Asl.Column,
873                         Op->Asl.Filename, MsgBuffer);
874     }
875     else
876     {
877         AslCommonError (ASL_ERROR, ASL_MSG_CORE_EXCEPTION, 0,
878                         0, 0, 0, NULL, MsgBuffer);
879     }

881     if (Abort)
882     {
883         AslAbort ();
884     }
885 }


888 /*******************************************************************************
889  *
890  * FUNCTION:    AslCompilererror
891  *
892  * PARAMETERS:  CompilerMessage         - Error message from the parser
893  *
894  * RETURN:      Status (0 for now)
895  *
896  * DESCRIPTION: Report an error situation discovered in a production
897  *              NOTE: don't change the name of this function, it is called
898  *              from the auto-generated parser.
899  *
900  ******************************************************************************/

902 int
903 AslCompilererror (
904     const char              *CompilerMessage)
905 {

907     AslCommonError (ASL_ERROR, ASL_MSG_SYNTAX, Gbl_CurrentLineNumber,
908         Gbl_LogicalLineNumber, Gbl_CurrentLineOffset,
909         Gbl_CurrentColumn, Gbl_Files[ASL_FILE_INPUT].Filename,
910         ACPI_CAST_PTR (char, CompilerMessage));

912     return (0);
913 }
```

```
**********************************************************
   10151 Thu Dec 26 13:48:27 2013
new/usr/src/common/acpica/compiler/aslfileio.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslfileio - File I/O support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslfileio")


/*******************************************************************************
 *
 * FUNCTION:    AslAbort
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the error log and abort the compiler. Used for serious
 *              I/O errors.
 *
 ******************************************************************************/
```



```
void
AslAbort (
    void)
{

    AePrintErrorLog (ASL_FILE_STDERR);
    if (Gbl_DebugFlag)
    {
        /* Print error summary to stdout also */

        AePrintErrorLog (ASL_FILE_STDOUT);
    }

    exit (1);
}


/*******************************************************************************
 *
 * FUNCTION:    FlFileError
 *
 * PARAMETERS:  FileId              - Index into file info array
 *              ErrorId             - Index into error message array
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode errno to an error message and add the entire error
 *              to the error log.
 *
 ******************************************************************************/

void
FlFileError (
    UINT32                  FileId,
    UINT8                   ErrorId)
{

    sprintf (MsgBuffer, "\"%s\" (%s)", Gbl_Files[FileId].Filename,
        strerror (errno));
    AslCommonError (ASL_ERROR, ErrorId, 0, 0, 0, 0, NULL, MsgBuffer);
}


/*******************************************************************************
 *
 * FUNCTION:    FlOpenFile
 *
 * PARAMETERS:  FileId              - Index into file info array
 *              Filename            - file pathname to open
 *              Mode                - Open mode for fopen
 *
 * RETURN:      None
 *
 * DESCRIPTION: Open a file.
 *              NOTE: Aborts compiler on any error.
 *
 ******************************************************************************/

void
FlOpenFile (
    UINT32                  FileId,
    char                    *Filename,
    char                    *Mode)
{
    FILE                    *File;
```

```
130     File = fopen (Filename, Mode);
131     if (!File)
132     {
133         FlFileError (FileId, ASL_MSG_OPEN);
134         AslAbort ();
135     }

137     Gbl_Files[FileId].Filename = Filename;
138     Gbl_Files[FileId].Handle   = File;
139 }


142 /******************************************************************************
143  *
144  * FUNCTION:    FlGetFileSize
145  *
146  * PARAMETERS:  FileId              - Index into file info array
147  *
148  * RETURN:      File Size
149  *
150  * DESCRIPTION: Get current file size. Uses seek-to-EOF. File must be open.
151  *
152  *****************************************************************************/

154 UINT32
155 FlGetFileSize (
156     UINT32                  FileId)
157 {
158     FILE                    *fp;
159     UINT32                  FileSize;
160     long                    Offset;


163     fp = Gbl_Files[FileId].Handle;
164     Offset = ftell (fp);

166     fseek (fp, 0, SEEK_END);
167     FileSize = (UINT32) ftell (fp);

169     /* Restore file pointer */

171     fseek (fp, Offset, SEEK_SET);
172     return (FileSize);
173 }


176 /******************************************************************************
177  *
178  * FUNCTION:    FlReadFile
179  *
180  * PARAMETERS:  FileId              - Index into file info array
181  *              Buffer              - Where to place the data
182  *              Length              - Amount to read
183  *
184  * RETURN:      Status. AE_ERROR indicates EOF.
185  *
186  * DESCRIPTION: Read data from an open file.
187  *              NOTE: Aborts compiler on any error.
188  *
189  *****************************************************************************/

191 ACPI_STATUS
192 FlReadFile (
193     UINT32                  FileId,
```



```
194     void                    *Buffer,
195     UINT32                  Length)
196 {
197     UINT32                  Actual;


200     /* Read and check for error */

202     Actual = fread (Buffer, 1, Length, Gbl_Files[FileId].Handle);
203     if (Actual < Length)
204     {
205         if (feof (Gbl_Files[FileId].Handle))
206         {
207             /* End-of-file, just return error */

209             return (AE_ERROR);
210         }

212         FlFileError (FileId, ASL_MSG_READ);
213         AslAbort ();
214     }

216     return (AE_OK);
217 }


220 /******************************************************************************
221  *
222  * FUNCTION:    FlWriteFile
223  *
224  * PARAMETERS:  FileId              - Index into file info array
225  *              Buffer              - Data to write
226  *              Length              - Amount of data to write
227  *
228  * RETURN:      None
229  *
230  * DESCRIPTION: Write data to an open file.
231  *              NOTE: Aborts compiler on any error.
232  *
233  *****************************************************************************/

235 void
236 FlWriteFile (
237     UINT32                  FileId,
238     void                    *Buffer,
239     UINT32                  Length)
240 {
241     UINT32                  Actual;


244     /* Write and check for error */

246     Actual = fwrite ((char *) Buffer, 1, Length, Gbl_Files[FileId].Handle);
247     if (Actual != Length)
248     {
249         FlFileError (FileId, ASL_MSG_WRITE);
250         AslAbort ();
251     }
252 }


255 /******************************************************************************
256  *
257  * FUNCTION:    FlPrintFile
258  *
259  * PARAMETERS:  FileId              - Index into file info array
```

```
 *              Format              - Printf format string
 *              ...                 - Printf arguments
 *
 * RETURN:      None
 *
 * DESCRIPTION: Formatted write to an open file.
 *              NOTE: Aborts compiler on any error.
 *
 ******************************************************************************/

void
FlPrintFile (
    UINT32                  FileId,
    char                    *Format,
    ...)
{
    INT32                   Actual;
    va_list                 Args;


    va_start (Args, Format);

    Actual = vfprintf (Gbl_Files[FileId].Handle, Format, Args);
    va_end (Args);

    if (Actual == -1)
    {
        FlFileError (FileId, ASL_MSG_WRITE);
        AslAbort ();
    }
}


/******************************************************************************
 *
 * FUNCTION:    FlSeekFile
 *
 * PARAMETERS:  FileId              - Index into file info array
 *              Offset              - Absolute byte offset in file
 *
 * RETURN:      None
 *
 * DESCRIPTION: Seek to absolute offset.
 *              NOTE: Aborts compiler on any error.
 *
 ******************************************************************************/

void
FlSeekFile (
    UINT32                  FileId,
    long                    Offset)
{
    int                     Error;


    Error = fseek (Gbl_Files[FileId].Handle, Offset, SEEK_SET);
    if (Error)
    {
        FlFileError (FileId, ASL_MSG_SEEK);
        AslAbort ();
    }
}


/******************************************************************************
 *
```



```
 * FUNCTION:    FlCloseFile
 *
 * PARAMETERS:  FileId              - Index into file info array
 *
 * RETURN:      None
 *
 * DESCRIPTION: Close an open file. Aborts compiler on error
 *
 ******************************************************************************/

void
FlCloseFile (
    UINT32                  FileId)
{
    int                     Error;


    if (!Gbl_Files[FileId].Handle)
    {
        return;
    }

    Error = fclose (Gbl_Files[FileId].Handle);
    if (Error)
    {
        FlFileError (FileId, ASL_MSG_CLOSE);
        AslAbort ();
    }

    Gbl_Files[FileId].Handle = NULL;
    return;
}


/******************************************************************************
 *
 * FUNCTION:    FlDeleteFile
 *
 * PARAMETERS:  FileId              - Index into file info array
 *
 * RETURN:      None
 *
 * DESCRIPTION: Delete a file.
 *
 ******************************************************************************/

void
FlDeleteFile (
    UINT32                  FileId)
{
    ASL_FILE_INFO           *Info = &Gbl_Files[FileId];


    if (!Info->Filename)
    {
        return;
    }

    if (remove (Info->Filename))
    {
        printf ("%s (%s file) ",
            Info->Filename, Info->Description);
        perror ("Could not delete");
    }

    Info->Filename = NULL;
```

```
392     return;
393 }
```

```
**********************************************************
   23426 Thu Dec 26 13:48:28 2013
new/usr/src/common/acpica/compiler/aslfiles.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslfiles - File support functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "acapps.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslfiles")

/* Local prototypes */

FILE *
FlOpenIncludeWithPrefix (
    char                    *PrefixDir,
    char                    *Filename);


#ifdef ACPI_OBSOLETE_FUNCTIONS
ACPI_STATUS
FlParseInputPathname (
    char                    *InputFilename);
#endif


/******************************************************************************
 *
 * FUNCTION:    FlSetLineNumber
 *
 * PARAMETERS:  Op        - Parse node for the LINE asl statement
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Set the current line number
 *
 *****************************************************************************/

void
FlSetLineNumber (
    UINT32                  LineNumber)
{

    DbgPrint (ASL_PARSE_OUTPUT, "\n#line: New line number %u (old %u)\n",
        LineNumber, Gbl_LogicalLineNumber);

    Gbl_CurrentLineNumber = LineNumber;
    Gbl_LogicalLineNumber = LineNumber;
}


/******************************************************************************
 *
 * FUNCTION:    FlSetFilename
 *
 * PARAMETERS:  Op        - Parse node for the LINE asl statement
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Set the current filename
 *
 *****************************************************************************/

void
FlSetFilename (
    char                    *Filename)
{

    DbgPrint (ASL_PARSE_OUTPUT, "\n#line: New filename %s (old %s)\n",
        Filename, Gbl_Files[ASL_FILE_INPUT].Filename);

    Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
}


/******************************************************************************
 *
 * FUNCTION:    FlAddIncludeDirectory
 *
 * PARAMETERS:  Dir             - Directory pathname string
 *
 * RETURN:      None
 *
 * DESCRIPTION: Add a directory the list of include prefix directories.
 *
 *****************************************************************************/
```

```
126 void
127 FlAddIncludeDirectory (
128     char                    *Dir)
129 {
130     ASL_INCLUDE_DIR         *NewDir;
131     ASL_INCLUDE_DIR         *NextDir;
132     ASL_INCLUDE_DIR         *PrevDir = NULL;
133     UINT32                  NeedsSeparator = 0;
134     size_t                  DirLength;


137     DirLength = strlen (Dir);
138     if (!DirLength)
139     {
140         return;
141     }

143     /* Make sure that the pathname ends with a path separator */

145     if ((Dir[DirLength-1] != '/') &&
146         (Dir[DirLength-1] != '\\'))
147     {
148         NeedsSeparator = 1;
149     }

151     NewDir = ACPI_ALLOCATE_ZEROED (sizeof (ASL_INCLUDE_DIR));
152     NewDir->Dir = ACPI_ALLOCATE (DirLength + 1 + NeedsSeparator);
153     strcpy (NewDir->Dir, Dir);
154     if (NeedsSeparator)
155     {
156         strcat (NewDir->Dir, "/");
157     }

159     /*
160      * Preserve command line ordering of -I options by adding new elements
161      * at the end of the list
162      */
163     NextDir = Gbl_IncludeDirList;
164     while (NextDir)
165     {
166         PrevDir = NextDir;
167         NextDir = NextDir->Next;
168     }

170     if (PrevDir)
171     {
172         PrevDir->Next = NewDir;
173     }
174     else
175     {
176         Gbl_IncludeDirList = NewDir;
177     }
178 }


181 /******************************************************************************
182  *
183  * FUNCTION:    FlMergePathnames
184  *
185  * PARAMETERS:  PrefixDir       - Prefix directory pathname. Can be NULL or
186  *                                a zero length string.
187  *              FilePathname    - The include filename from the source ASL.
188  *
189  * RETURN:      Merged pathname string
190  *
191  * DESCRIPTION: Merge two pathnames that (probably) have common elements, to
```



```
192  *              arrive at a minimal length string. Merge can occur if the
193  *              FilePathname is relative to the PrefixDir.
194  *
195  *****************************************************************************/

197 char *
198 FlMergePathnames (
199     char                    *PrefixDir,
200     char                    *FilePathname)
201 {
202     char                    *CommonPath;
203     char                    *Pathname;
204     char                    *LastElement;


207     DbgPrint (ASL_PARSE_OUTPUT, "Include: Prefix path - \"%s\"\n"
208         "Include: FilePathname - \"%s\"\n",
209          PrefixDir, FilePathname);

211     /*
212      * If there is no prefix directory or if the file pathname is absolute,
213      * just return the original file pathname
214      */
215     if (!PrefixDir || (!*PrefixDir) ||
216         (*FilePathname == '/') ||
217          (FilePathname[1] == ':'))
218     {
219         Pathname = ACPI_ALLOCATE (strlen (FilePathname) + 1);
220         strcpy (Pathname, FilePathname);
221         goto ConvertBackslashes;
222     }

224     /* Need a local copy of the prefix directory path */

226     CommonPath = ACPI_ALLOCATE (strlen (PrefixDir) + 1);
227     strcpy (CommonPath, PrefixDir);

229     /*
230      * Walk forward through the file path, and simultaneously backward
231      * through the prefix directory path until there are no more
232      * relative references at the start of the file path.
233      */
234     while (*FilePathname && (!strncmp (FilePathname, "../", 3)))
235     {
236         /* Remove last element of the prefix directory path */

238         LastElement = strrchr (CommonPath, '/');
239         if (!LastElement)
240         {
241             goto ConcatenatePaths;
242         }

244         *LastElement = 0;   /* Terminate CommonPath string */
245         FilePathname += 3;  /* Point to next path element */
246     }

248     /*
249      * Remove the last element of the prefix directory path (it is the same as
250      * the first element of the file pathname), and build the final merged
251      * pathname.
252      */
253     LastElement = strrchr (CommonPath, '/');
254     if (LastElement)
255     {
256         *LastElement = 0;
257     }
```

```
259     /* Build the final merged pathname */

261 ConcatenatePaths:
262     Pathname = ACPI_ALLOCATE_ZEROED (strlen (CommonPath) + strlen (FilePathname)
263     if (LastElement && *CommonPath)
264     {
265         strcpy (Pathname, CommonPath);
266         strcat (Pathname, "/");
267     }
268     strcat (Pathname, FilePathname);
269     ACPI_FREE (CommonPath);

271     /* Convert all backslashes to normal slashes */

273 ConvertBackslashes:
274     UtConvertBackslashes (Pathname);

276     DbgPrint (ASL_PARSE_OUTPUT, "Include: Merged Pathname - \"%s\"\n",
277          Pathname);
278     return (Pathname);
279 }


282 /******************************************************************************
283  *
284  * FUNCTION:    FlOpenIncludeWithPrefix
285  *
286  * PARAMETERS:  PrefixDir       - Prefix directory pathname. Can be a zero
287  *                                length string.
288  *              Filename        - The include filename from the source ASL.
289  *
290  * RETURN:      Valid file descriptor if successful. Null otherwise.
291  *
292  * DESCRIPTION: Open an include file and push it on the input file stack.
293  *
294  *****************************************************************************/

296 FILE *
297 FlOpenIncludeWithPrefix (
298     char                    *PrefixDir,
299     char                    *Filename)
300 {
301     FILE                    *IncludeFile;
302     char                    *Pathname;


305     /* Build the full pathname to the file */

307     Pathname = FlMergePathnames (PrefixDir, Filename);

309     DbgPrint (ASL_PARSE_OUTPUT, "Include: Opening file - \"%s\"\n\n",
310         Pathname);

312     /* Attempt to open the file, push if successful */

314     IncludeFile = fopen (Pathname, "r");
315     if (!IncludeFile)
316     {
317         fprintf (stderr, "Could not open include file %s\n", Pathname);
318         ACPI_FREE (Pathname);
319         return (NULL);
320     }

322     /* Push the include file on the open input file stack */
```



```
324     AslPushInputFileStack (IncludeFile, Pathname);
325     return (IncludeFile);
326 }


329 /******************************************************************************
330  *
331  * FUNCTION:    FlOpenIncludeFile
332  *
333  * PARAMETERS:  Op          - Parse node for the INCLUDE ASL statement
334  *
335  * RETURN:      None.
336  *
337  * DESCRIPTION: Open an include file and push it on the input file stack.
338  *
339  *****************************************************************************/

341 void
342 FlOpenIncludeFile (
343     ACPI_PARSE_OBJECT       *Op)
344 {
345     FILE                    *IncludeFile;
346     ASL_INCLUDE_DIR         *NextDir;


349     /* Op must be valid */

351     if (!Op)
352     {
353         AslCommonError (ASL_ERROR, ASL_MSG_INCLUDE_FILE_OPEN,
354             Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
355             Gbl_InputByteCount, Gbl_CurrentColumn,
356             Gbl_Files[ASL_FILE_INPUT].Filename, " - Null parse node");

358         return;
359     }

361     /*
362      * Flush out the "include ()" statement on this line, start
363      * the actual include file on the next line
364      */
365     AslResetCurrentLineBuffer ();
366     FlPrintFile (ASL_FILE_SOURCE_OUTPUT, "\n");
367     Gbl_CurrentLineOffset++;


370     /* Attempt to open the include file */

372     /* If the file specifies an absolute path, just open it */

374     if ((Op->Asl.Value.String[0] == '/')  ||
375         (Op->Asl.Value.String[0] == '\\') ||
376         (Op->Asl.Value.String[1] == ':'))
377     {
378         IncludeFile = FlOpenIncludeWithPrefix ("", Op->Asl.Value.String);
379         if (!IncludeFile)
380         {
381             goto ErrorExit;
382         }
383         return;
384     }

386     /*
387      * The include filename is not an absolute path.
388      *
389      * First, search for the file within the "local" directory -- meaning
```

```
390      * the same directory that contains the source file.
391      *
392      * Construct the file pathname from the global directory name.
393      */
394     IncludeFile = FlOpenIncludeWithPrefix (Gbl_DirectoryPath, Op->Asl.Value.Stri
395     if (IncludeFile)
396     {
397         return;
398     }

400     /*
401      * Second, search for the file within the (possibly multiple) directories
402      * specified by the -I option on the command line.
403      */
404     NextDir = Gbl_IncludeDirList;
405     while (NextDir)
406     {
407         IncludeFile = FlOpenIncludeWithPrefix (NextDir->Dir, Op->Asl.Value.Strin
408         if (IncludeFile)
409         {
410             return;
411         }

413         NextDir = NextDir->Next;
414     }

416     /* We could not open the include file after trying very hard */

418 ErrorExit:
419     sprintf (MsgBuffer, "%s, %s", Op->Asl.Value.String, strerror (errno));
420     AslError (ASL_ERROR, ASL_MSG_INCLUDE_FILE_OPEN, Op, MsgBuffer);
421 }


424 /******************************************************************************
425  *
426  * FUNCTION:    FlOpenInputFile
427  *
428  * PARAMETERS:  InputFilename       - The user-specified ASL source file to be
429  *                                    compiled
430  *
431  * RETURN:      Status
432  *
433  * DESCRIPTION: Open the specified input file, and save the directory path to
434  *              the file so that include files can be opened in
435  *              the same directory.
436  *
437  *****************************************************************************/

439 ACPI_STATUS
440 FlOpenInputFile (
441     char                    *InputFilename)
442 {

444     /* Open the input ASL file, text mode */

446     FlOpenFile (ASL_FILE_INPUT, InputFilename, "rt");
447     AslCompilerin = Gbl_Files[ASL_FILE_INPUT].Handle;

449     return (AE_OK);
450 }


453 /******************************************************************************
454  *
455  * FUNCTION:    FlOpenAmlOutputFile
```



```
456  *
457  * PARAMETERS:  FilenamePrefix       - The user-specified ASL source file
458  *
459  * RETURN:      Status
460  *
461  * DESCRIPTION: Create the output filename (*.AML) and open the file. The file
462  *              is created in the same directory as the parent input file.
463  *
464  *****************************************************************************/

466 ACPI_STATUS
467 FlOpenAmlOutputFile (
468     char                    *FilenamePrefix)
469 {
470     char                    *Filename;


473     /* Output filename usually comes from the ASL itself */

475     Filename = Gbl_Files[ASL_FILE_AML_OUTPUT].Filename;
476     if (!Filename)
477     {
478         /* Create the output AML filename */

480         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_AML_CODE);
481         if (!Filename)
482         {
483             AslCommonError (ASL_ERROR, ASL_MSG_OUTPUT_FILENAME,
484                 0, 0, 0, 0, NULL, NULL);
485             return (AE_ERROR);
486         }
487     }

489     /* Open the output AML file in binary mode */

491     FlOpenFile (ASL_FILE_AML_OUTPUT, Filename, "w+b");
492     return (AE_OK);
493 }


496 /******************************************************************************
497  *
498  * FUNCTION:    FlOpenMiscOutputFiles
499  *
500  * PARAMETERS:  FilenamePrefix       - The user-specified ASL source file
501  *
502  * RETURN:      Status
503  *
504  * DESCRIPTION: Create and open the various output files needed, depending on
505  *              the command line options
506  *
507  *****************************************************************************/

509 ACPI_STATUS
510 FlOpenMiscOutputFiles (
511     char                    *FilenamePrefix)
512 {
513     char                    *Filename;


516     /* All done for disassembler */

518     if (Gbl_FileType == ASL_INPUT_TYPE_ACPI_TABLE)
519     {
520         return (AE_OK);
521     }
```

```
523     /* Create/Open a hex output file if asked */
525     if (Gbl_HexOutputFlag)
526     {
527         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_HEX_DUMP);
528         if (!Filename)
529         {
530             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
531                 0, 0, 0, 0, NULL, NULL);
532             return (AE_ERROR);
533         }
535         /* Open the hex file, text mode */
537         FlOpenFile (ASL_FILE_HEX_OUTPUT, Filename, "w+t");
539         AslCompilerSignon (ASL_FILE_HEX_OUTPUT);
540         AslCompilerFileHeader (ASL_FILE_HEX_OUTPUT);
541     }
543     /* Create/Open a debug output file if asked */
545     if (Gbl_DebugFlag)
546     {
547         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_DEBUG);
548         if (!Filename)
549         {
550             AslCommonError (ASL_ERROR, ASL_MSG_DEBUG_FILENAME,
551                 0, 0, 0, 0, NULL, NULL);
552             return (AE_ERROR);
553         }
555         /* Open the debug file as STDERR, text mode */
557         /* TBD: hide this behind a FlReopenFile function */
559         Gbl_Files[ASL_FILE_DEBUG_OUTPUT].Filename = Filename;
560         Gbl_Files[ASL_FILE_DEBUG_OUTPUT].Handle =
561             freopen (Filename, "w+t", stderr);
563         if (!Gbl_Files[ASL_FILE_DEBUG_OUTPUT].Handle)
564         {
565             AslCommonError (ASL_ERROR, ASL_MSG_DEBUG_FILENAME,
566                 0, 0, 0, 0, NULL, NULL);
567             return (AE_ERROR);
568         }
570         AslCompilerSignon (ASL_FILE_DEBUG_OUTPUT);
571         AslCompilerFileHeader (ASL_FILE_DEBUG_OUTPUT);
572     }
574     /* Create/Open a listing output file if asked */
576     if (Gbl_ListingFlag)
577     {
578         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_LISTING);
579         if (!Filename)
580         {
581             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
582                 0, 0, 0, 0, NULL, NULL);
583             return (AE_ERROR);
584         }
586         /* Open the listing file, text mode */
```



```
588         FlOpenFile (ASL_FILE_LISTING_OUTPUT, Filename, "w+t");
590         AslCompilerSignon (ASL_FILE_LISTING_OUTPUT);
591         AslCompilerFileHeader (ASL_FILE_LISTING_OUTPUT);
592     }
594     /* Create the preprocessor output file if preprocessor enabled */
596     if (Gbl_PreprocessFlag)
597     {
598         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_PREPROCESSOR)
599         if (!Filename)
600         {
601             AslCommonError (ASL_ERROR, ASL_MSG_PREPROCESSOR_FILENAME,
602                 0, 0, 0, 0, NULL, NULL);
603             return (AE_ERROR);
604         }
606         FlOpenFile (ASL_FILE_PREPROCESSOR, Filename, "w+t");
607     }
609     /* All done for data table compiler */
611     if (Gbl_FileType == ASL_INPUT_TYPE_ASCII_DATA)
612     {
613         return (AE_OK);
614     }
616     /* Create/Open a combined source output file */
618     Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_SOURCE);
619     if (!Filename)
620     {
621         AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
622             0, 0, 0, 0, NULL, NULL);
623         return (AE_ERROR);
624     }
626     /*
627      * Open the source output file, binary mode (so that LF does not get
628      * expanded to CR/LF on some systems, messing up our seek
629      * calculations.)
630      */
631     FlOpenFile (ASL_FILE_SOURCE_OUTPUT, Filename, "w+b");
633 /*
634 // TBD: TEMP
635 //    AslCompilerin = Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Handle;
636 */
637     /* Create/Open a assembly code source output file if asked */
639     if (Gbl_AsmOutputFlag)
640     {
641         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_ASM_SOURCE);
642         if (!Filename)
643         {
644             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
645                 0, 0, 0, 0, NULL, NULL);
646             return (AE_ERROR);
647         }
649         /* Open the assembly code source file, text mode */
651         FlOpenFile (ASL_FILE_ASM_SOURCE_OUTPUT, Filename, "w+t");
653         AslCompilerSignon (ASL_FILE_ASM_SOURCE_OUTPUT);
```

```
654         AslCompilerFileHeader (ASL_FILE_ASM_SOURCE_OUTPUT);
655     }
657     /* Create/Open a C code source output file if asked */
659     if (Gbl_C_OutputFlag)
660     {
661         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_C_SOURCE);
662         if (!Filename)
663         {
664             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
665                 0, 0, 0, 0, NULL, NULL);
666             return (AE_ERROR);
667         }
669         /* Open the C code source file, text mode */
671         FlOpenFile (ASL_FILE_C_SOURCE_OUTPUT, Filename, "w+t");
673         FlPrintFile (ASL_FILE_C_SOURCE_OUTPUT, "/*\n");
674         AslCompilerSignon (ASL_FILE_C_SOURCE_OUTPUT);
675         AslCompilerFileHeader (ASL_FILE_C_SOURCE_OUTPUT);
676     }
678     /* Create/Open a C code source output file for the offset table if asked */
680     if (Gbl_C_OffsetTableFlag)
681     {
682         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_C_OFFSET);
683         if (!Filename)
684         {
685             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
686                 0, 0, 0, 0, NULL, NULL);
687             return (AE_ERROR);
688         }
690         /* Open the C code source file, text mode */
692         FlOpenFile (ASL_FILE_C_OFFSET_OUTPUT, Filename, "w+t");
694         FlPrintFile (ASL_FILE_C_OFFSET_OUTPUT, "/*\n");
695         AslCompilerSignon (ASL_FILE_C_OFFSET_OUTPUT);
696         AslCompilerFileHeader (ASL_FILE_C_OFFSET_OUTPUT);
697     }
699     /* Create/Open a assembly include output file if asked */
701     if (Gbl_AsmIncludeOutputFlag)
702     {
703         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_ASM_INCLUDE);
704         if (!Filename)
705         {
706             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
707                 0, 0, 0, 0, NULL, NULL);
708             return (AE_ERROR);
709         }
711         /* Open the assembly include file, text mode */
713         FlOpenFile (ASL_FILE_ASM_INCLUDE_OUTPUT, Filename, "w+t");
715         AslCompilerSignon (ASL_FILE_ASM_INCLUDE_OUTPUT);
716         AslCompilerFileHeader (ASL_FILE_ASM_INCLUDE_OUTPUT);
717     }
719     /* Create/Open a C include output file if asked */
```



```
721     if (Gbl_C_IncludeOutputFlag)
722     {
723         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_C_INCLUDE);
724         if (!Filename)
725         {
726             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
727                 0, 0, 0, 0, NULL, NULL);
728             return (AE_ERROR);
729         }
731         /* Open the C include file, text mode */
733         FlOpenFile (ASL_FILE_C_INCLUDE_OUTPUT, Filename, "w+t");
735         FlPrintFile (ASL_FILE_C_INCLUDE_OUTPUT, "/*\n");
736         AslCompilerSignon (ASL_FILE_C_INCLUDE_OUTPUT);
737         AslCompilerFileHeader (ASL_FILE_C_INCLUDE_OUTPUT);
738     }
740     /* Create a namespace output file if asked */
742     if (Gbl_NsOutputFlag)
743     {
744         Filename = FlGenerateFilename (FilenamePrefix, FILE_SUFFIX_NAMESPACE);
745         if (!Filename)
746         {
747             AslCommonError (ASL_ERROR, ASL_MSG_LISTING_FILENAME,
748                 0, 0, 0, 0, NULL, NULL);
749             return (AE_ERROR);
750         }
752         /* Open the namespace file, text mode */
754         FlOpenFile (ASL_FILE_NAMESPACE_OUTPUT, Filename, "w+t");
756         AslCompilerSignon (ASL_FILE_NAMESPACE_OUTPUT);
757         AslCompilerFileHeader (ASL_FILE_NAMESPACE_OUTPUT);
758     }
760     return (AE_OK);
761 }


764 #ifdef ACPI_OBSOLETE_FUNCTIONS
765 /*******************************************************************************
766  *
767  * FUNCTION:    FlParseInputPathname
768  *
769  * PARAMETERS:  InputFilename       - The user-specified ASL source file to be
770  *                                    compiled
771  *
772  * RETURN:      Status
773  *
774  * DESCRIPTION: Split the input path into a directory and filename part
775  *              1) Directory part used to open include files
776  *              2) Filename part used to generate output filenames
777  *
778  ******************************************************************************/
780 ACPI_STATUS
781 FlParseInputPathname (
782     char                    *InputFilename)
783 {
784     char                    *Substring;
```

```
787     if (!InputFilename)
788     {
789         return (AE_OK);
790     }

792     /* Get the path to the input filename's directory */

794     Gbl_DirectoryPath = strdup (InputFilename);
795     if (!Gbl_DirectoryPath)
796     {
797         return (AE_NO_MEMORY);
798     }

800     Substring = strrchr (Gbl_DirectoryPath, '\\');
801     if (!Substring)
802     {
803         Substring = strrchr (Gbl_DirectoryPath, '/');
804         if (!Substring)
805         {
806             Substring = strrchr (Gbl_DirectoryPath, ':');
807         }
808     }

810     if (!Substring)
811     {
812         Gbl_DirectoryPath[0] = 0;
813         if (Gbl_UseDefaultAmlFilename)
814         {
815             Gbl_OutputFilenamePrefix = strdup (InputFilename);
816         }
817     }
818     else
819     {
820         if (Gbl_UseDefaultAmlFilename)
821         {
822             Gbl_OutputFilenamePrefix = strdup (Substring + 1);
823         }
824         *(Substring+1) = 0;
825     }

827     UtConvertBackslashes (Gbl_OutputFilenamePrefix);
828     return (AE_OK);
829 }
830 #endif
```

```
**********************************************************
   15815 Thu Dec 26 13:48:28 2013
new/usr/src/common/acpica/compiler/aslfold.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslfold - Constant folding
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#include "acdispat.h"
#include "acparser.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslfold")

/* Local prototypes */

static ACPI_STATUS
OpcAmlEvaluationWalk1 (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
OpcAmlEvaluationWalk2 (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
OpcAmlCheckForConstant (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static void
OpcUpdateIntegerNode (
    ACPI_PARSE_OBJECT       *Op,
    UINT64                  Value);


/******************************************************************************
 *
 * FUNCTION:    OpcAmlEvaluationWalk1
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Descending callback for AML execution of constant subtrees
 *
 *****************************************************************************/

static ACPI_STATUS
OpcAmlEvaluationWalk1 (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_WALK_STATE         *WalkState = Context;
    ACPI_STATUS             Status;
    ACPI_PARSE_OBJECT       *OutOp;


    WalkState->Op = Op;
    WalkState->Opcode = Op->Common.AmlOpcode;
    WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    /* Copy child pointer to Arg for compatibility with Interpreter */

    if (Op->Asl.Child)
    {
        Op->Common.Value.Arg = Op->Asl.Child;
    }

    /* Call AML dispatcher */

    Status = AcpiDsExecBeginOp (WalkState, &OutOp);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("Constant interpretation failed - %s\n",
                      AcpiFormatException (Status));
    }

    return (Status);
}
```

```
/*******************************************************************************
 *
 * FUNCTION:    OpcAmlEvaluationWalk2
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Ascending callback for AML execution of constant subtrees
 *
 ******************************************************************************/

static ACPI_STATUS
OpcAmlEvaluationWalk2 (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_WALK_STATE         *WalkState = Context;
    ACPI_STATUS             Status;


    WalkState->Op = Op;
    WalkState->Opcode = Op->Common.AmlOpcode;
    WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    /* Copy child pointer to Arg for compatibility with Interpreter */

    if (Op->Asl.Child)
    {
        Op->Common.Value.Arg = Op->Asl.Child;
    }

    /* Call AML dispatcher */

    Status = AcpiDsExecEndOp (WalkState);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("Constant interpretation failed - %s\n",
                      AcpiFormatException (Status));
    }

    return (Status);
}


/*******************************************************************************
 *
 * FUNCTION:    OpcAmlCheckForConstant
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check one Op for a type 3/4/5 AML opcode
 *
 ******************************************************************************/

static ACPI_STATUS
OpcAmlCheckForConstant (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_WALK_STATE         *WalkState = Context;
```



```


    WalkState->Op = Op;
    WalkState->Opcode = Op->Common.AmlOpcode;
    WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    DbgPrint (ASL_PARSE_OUTPUT, "[%.4d] Opcode: %12.12s ",
                Op->Asl.LogicalLineNumber, Op->Asl.ParseOpName);

    /*
     * These opcodes do not appear in the OpcodeInfo table, but
     * they represent constants, so abort the constant walk now.
     */
    if ((WalkState->Opcode == AML_RAW_DATA_BYTE) ||
        (WalkState->Opcode == AML_RAW_DATA_WORD) ||
        (WalkState->Opcode == AML_RAW_DATA_DWORD) ||
        (WalkState->Opcode == AML_RAW_DATA_QWORD))
    {
        WalkState->WalkType = ACPI_WALK_CONST_OPTIONAL;
        return (AE_TYPE);
    }

    if (!(WalkState->OpInfo->Flags & AML_CONSTANT))
    {
        /* The opcode is not a Type 3/4/5 opcode */

        if (Op->Asl.CompileFlags & NODE_IS_TARGET)
        {
            DbgPrint (ASL_PARSE_OUTPUT,
                "**** Valid Target, cannot reduce ****\n");
        }
        else
        {
            DbgPrint (ASL_PARSE_OUTPUT,
                "**** Not a Type 3/4/5 opcode ****\n");
        }

        if (WalkState->WalkType == ACPI_WALK_CONST_OPTIONAL)
        {
            /*
             * We are looking at at normal expression to see if it can be
             * reduced. It can't. No error
             */
            return (AE_TYPE);
        }

        /*
         * This is an expression that MUST reduce to a constant, and it
         * can't be reduced. This is an error
         */
        if (Op->Asl.CompileFlags & NODE_IS_TARGET)
        {
            AslError (ASL_ERROR, ASL_MSG_INVALID_TARGET, Op,
                Op->Asl.ParseOpName);
        }
        else
        {
            AslError (ASL_ERROR, ASL_MSG_INVALID_CONSTANT_OP, Op,
                Op->Asl.ParseOpName);
        }

        return (AE_TYPE);
    }

    /* Debug output */
```

```
259     DbgPrint (ASL_PARSE_OUTPUT, "TYPE_345");
260
261     if (Op->Asl.CompileFlags & NODE_IS_TARGET)
262     {
263         DbgPrint (ASL_PARSE_OUTPUT, " TARGET");
264     }
265     if (Op->Asl.CompileFlags & NODE_IS_TERM_ARG)
266     {
267         DbgPrint (ASL_PARSE_OUTPUT, " TERMARG");
268     }
269
270     DbgPrint (ASL_PARSE_OUTPUT, "\n");
271     return (AE_OK);
272 }
273
274
275 /*******************************************************************************
276  *
277  * FUNCTION:    OpcAmlConstantWalk
278  *
279  * PARAMETERS:  ASL_WALK_CALLBACK
280  *
281  * RETURN:      Status
282  *
283  * DESCRIPTION: Reduce an Op and its subtree to a constant if possible
284  *
285  ******************************************************************************/
286
287 ACPI_STATUS
288 OpcAmlConstantWalk (
289     ACPI_PARSE_OBJECT       *Op,
290     UINT32                  Level,
291     void                    *Context)
292 {
293     ACPI_WALK_STATE         *WalkState;
294     ACPI_STATUS             Status = AE_OK;
295     ACPI_OPERAND_OBJECT     *ObjDesc;
296     ACPI_PARSE_OBJECT       *RootOp;
297     ACPI_PARSE_OBJECT       *OriginalParentOp;
298     UINT8                   WalkType;
299
300
301     /*
302      * Only interested in subtrees that could possibly contain
303      * expressions that can be evaluated at this time
304      */
305     if ((!(Op->Asl.CompileFlags & NODE_COMPILE_TIME_CONST)) ||
306         (Op->Asl.CompileFlags & NODE_IS_TARGET))
307     {
308         return (AE_OK);
309     }
310
311     /* Set the walk type based on the reduction used for this op */
312
313     if (Op->Asl.CompileFlags & NODE_IS_TERM_ARG)
314     {
315         /* Op is a TermArg, constant folding is merely optional */
316
317         if (!Gbl_FoldConstants)
318         {
319             return (AE_CTRL_DEPTH);
320         }
321
322         WalkType = ACPI_WALK_CONST_OPTIONAL;
323     }
324     else
```



```
325     {
326         /* Op is a DataObject, the expression MUST reduced to a constant */
327
328         WalkType = ACPI_WALK_CONST_REQUIRED;
329     }
330
331     /* Create a new walk state */
332
333     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
334     if (!WalkState)
335     {
336         return (AE_NO_MEMORY);
337     }
338
339     WalkState->NextOp = NULL;
340     WalkState->Params = NULL;
341     WalkState->WalkType = WalkType;
342     WalkState->CallerReturnDesc = &ObjDesc;
343
344     /*
345      * Examine the entire subtree -- all nodes must be constants
346      * or type 3/4/5 opcodes
347      */
348     Status = TrWalkParseTree (Op, ASL_WALK_VISIT_DOWNWARD,
349         OpcAmlCheckForConstant, NULL, WalkState);
350
351     /*
352      * Did we find an entire subtree that contains all constants and type 3/4/5
353      * opcodes?  (Only AE_OK or AE_TYPE returned from above)
354      */
355     if (Status == AE_TYPE)
356     {
357         /* Subtree cannot be reduced to a constant */
358
359         if (WalkState->WalkType == ACPI_WALK_CONST_OPTIONAL)
360         {
361             AcpiDsDeleteWalkState (WalkState);
362             return (AE_OK);
363         }
364
365         /* Don't descend any further, and use a default "constant" value */
366
367         Status = AE_CTRL_DEPTH;
368     }
369     else
370     {
371         /* Subtree can be reduced */
372
373         /* Allocate a new temporary root for this subtree */
374
375         RootOp = TrAllocateNode (PARSEOP_INTEGER);
376         if (!RootOp)
377         {
378             return (AE_NO_MEMORY);
379         }
380
381         RootOp->Common.AmlOpcode = AML_INT_EVAL_SUBTREE_OP;
382
383         OriginalParentOp = Op->Common.Parent;
384         Op->Common.Parent = RootOp;
385
386         /* Hand off the subtree to the AML interpreter */
387
388         Status = TrWalkParseTree (Op, ASL_WALK_VISIT_TWICE,
389             OpcAmlEvaluationWalk1, OpcAmlEvaluationWalk2, WalkState);
390         Op->Common.Parent = OriginalParentOp;
```

```
392         /* TBD: we really *should* release the RootOp node */

394         if (ACPI_SUCCESS (Status))
395         {
396             TotalFolds++;

398             /* Get the final result */

400             Status = AcpiDsResultPop (&ObjDesc, WalkState);
401         }

403         /* Check for error from the ACPICA core */

405         if (ACPI_FAILURE (Status))
406         {
407             AslCoreSubsystemError (Op, Status,
408                 "Failure during constant evaluation", FALSE);
409         }
410     }

412     if (ACPI_FAILURE (Status))
413     {
414         /* We could not resolve the subtree for some reason */

416         AslError (ASL_ERROR, ASL_MSG_CONSTANT_EVALUATION, Op,
417             Op->Asl.ParseOpName);

419         /* Set the subtree value to ZERO anyway. Eliminates further errors */

421         OpcUpdateIntegerNode (Op, 0);
422     }
423     else
424     {
425         AslError (ASL_OPTIMIZATION, ASL_MSG_CONSTANT_FOLDED, Op,
426             Op->Asl.ParseOpName);

428         /*
429          * Because we know we executed type 3/4/5 opcodes above, we know that
430          * the result must be either an Integer, String, or Buffer.
431          */
432         switch (ObjDesc->Common.Type)
433         {
434         case ACPI_TYPE_INTEGER:

436             OpcUpdateIntegerNode (Op, ObjDesc->Integer.Value);

438             DbgPrint (ASL_PARSE_OUTPUT,
439                 "Constant expression reduced to (%s) %8.8X%8.8X\n",
440                 Op->Asl.ParseOpName,
441                 ACPI_FORMAT_UINT64 (Op->Common.Value.Integer));
442             break;

444         case ACPI_TYPE_STRING:

446             Op->Asl.ParseOpcode = PARSEOP_STRING_LITERAL;
447             Op->Common.AmlOpcode = AML_STRING_OP;
448             Op->Asl.AmlLength = ACPI_STRLEN (ObjDesc->String.Pointer) + 1;
449             Op->Common.Value.String = ObjDesc->String.Pointer;

451             DbgPrint (ASL_PARSE_OUTPUT,
452                 "Constant expression reduced to (STRING) %s\n",
453                 Op->Common.Value.String);

455             break;
```



```
457         case ACPI_TYPE_BUFFER:

459             Op->Asl.ParseOpcode = PARSEOP_BUFFER;
460             Op->Common.AmlOpcode = AML_BUFFER_OP;
461             Op->Asl.CompileFlags = NODE_AML_PACKAGE;
462             UtSetParseOpName (Op);

464             /* Child node is the buffer length */

466             RootOp = TrAllocateNode (PARSEOP_INTEGER);

468             RootOp->Asl.AmlOpcode = AML_DWORD_OP;
469             RootOp->Asl.Value.Integer = ObjDesc->Buffer.Length;
470             RootOp->Asl.Parent = Op;

472             (void) OpcSetOptimalIntegerSize (RootOp);

474             Op->Asl.Child = RootOp;
475             Op = RootOp;
476             UtSetParseOpName (Op);

478             /* Peer to the child is the raw buffer data */

480             RootOp = TrAllocateNode (PARSEOP_RAW_DATA);
481             RootOp->Asl.AmlOpcode = AML_RAW_DATA_BUFFER;
482             RootOp->Asl.AmlLength = ObjDesc->Buffer.Length;
483             RootOp->Asl.Value.String = (char *) ObjDesc->Buffer.Pointer;
484             RootOp->Asl.Parent = Op->Asl.Parent;

486             Op->Asl.Next = RootOp;
487             Op = RootOp;

489             DbgPrint (ASL_PARSE_OUTPUT,
490                 "Constant expression reduced to (BUFFER) length %X\n",
491                 ObjDesc->Buffer.Length);
492             break;

494         default:

496             printf ("Unsupported return type: %s\n",
497                 AcpiUtGetObjectTypeName (ObjDesc));
498             break;
499         }
500     }

502     UtSetParseOpName (Op);
503     Op->Asl.Child = NULL;

505     AcpiDsDeleteWalkState (WalkState);
506     return (AE_CTRL_DEPTH);
507 }


510 /*******************************************************************************
511  *
512  * FUNCTION:    OpcUpdateIntegerNode
513  *
514  * PARAMETERS:  Op                  - Current parse object
515  *
516  * RETURN:      None
517  *
518  * DESCRIPTION: Update node to the correct integer type.
519  *
520  ******************************************************************************/

522 static void
```

```
523 OpcUpdateIntegerNode (
524     ACPI_PARSE_OBJECT       *Op,
525     UINT64                  Value)
526 {

528     Op->Common.Value.Integer = Value;

530     /*
531      * The AmlLength is used by the parser to indicate a constant,
532      * (if non-zero). Length is either (1/2/4/8)
533      */
534     switch (Op->Asl.AmlLength)
535     {
536     case 1:

538         TrUpdateNode (PARSEOP_BYTECONST, Op);
539         Op->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
540         break;

542     case 2:

544         TrUpdateNode (PARSEOP_WORDCONST, Op);
545         Op->Asl.AmlOpcode = AML_RAW_DATA_WORD;
546         break;

548     case 4:

550         TrUpdateNode (PARSEOP_DWORDCONST, Op);
551         Op->Asl.AmlOpcode = AML_RAW_DATA_DWORD;
552         break;

554     case 8:

556         TrUpdateNode (PARSEOP_QWORDCONST, Op);
557         Op->Asl.AmlOpcode = AML_RAW_DATA_QWORD;
558         break;

560     case 0:
561     default:

563         OpcSetOptimalIntegerSize (Op);
564         TrUpdateNode (PARSEOP_INTEGER, Op);
565         break;
566     }

568     Op->Asl.AmlLength = 0;
569 }
```

```
**********************************************************
   12572 Thu Dec 26 13:48:28 2013
new/usr/src/common/acpica/compiler/aslglobal.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslglobal.h - Global variable definitions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ASLGLOBAL_H
#define __ASLGLOBAL_H


/*
 * Global variables. Defined in aslmain.c only, externed in all other files
 */

#undef ASL_EXTERN

#ifdef _DECLARE_GLOBALS
#define ASL_EXTERN
#define ASL_INIT_GLOBAL(a,b)        (a)=(b)
#else
#define ASL_EXTERN                  extern
```



```
#define ASL_INIT_GLOBAL(a,b)        (a)
#endif


#ifdef _DECLARE_GLOBALS
UINT32                              Gbl_ExceptionCount[ASL_NUM_REPORT_LEVELS] =
char                                AslHexLookup[] =
{
    '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
};


/* Table below must match ASL_FILE_TYPES in asltypes.h */

ASL_FILE_INFO                       Gbl_Files [ASL_NUM_FILES] =
{
    {NULL, NULL, "stdout:       ", "Standard Output"},
    {NULL, NULL, "stderr:       ", "Standard Error"},
    {NULL, NULL, "Table Input:  ", "Source Input"},
    {NULL, NULL, "Binary Output:", "AML Output"},
    {NULL, NULL, "Source Output:", "Source Output"},
    {NULL, NULL, "Preprocessor: ", "Preprocessor Output"},
    {NULL, NULL, "Listing File: ", "Listing Output"},
    {NULL, NULL, "Hex Dump:     ", "Hex Table Output"},
    {NULL, NULL, "Namespace:    ", "Namespace Output"},
    {NULL, NULL, "Debug File:   ", "Debug Output"},
    {NULL, NULL, "ASM Source:   ", "Assembly Code Output"},
    {NULL, NULL, "C Source:     ", "C Code Output"},
    {NULL, NULL, "ASM Include:  ", "Assembly Header Output"},
    {NULL, NULL, "C Include:    ", "C Header Output"},
    {NULL, NULL, "Offset Table: ", "C Offset Table Output"}
};

#else
extern UINT32                       Gbl_ExceptionCount[ASL_NUM_REPORT_LEVELS];
extern char                         AslHexLookup[];
extern ASL_FILE_INFO                Gbl_Files [ASL_NUM_FILES];
#endif


/*
 * Parser and other externals
 */
extern int                          yydebug;
extern FILE                         *AslCompilerin;
extern int                          AslCompilerdebug;
extern int                          DtParserdebug;
extern int                          PrParserdebug;
extern const ASL_MAPPING_ENTRY      AslKeywordMapping[];
extern char                         *AslCompilertext;

#define ASL_DEFAULT_LINE_BUFFER_SIZE    (1024 * 32) /* 32K */
#define ASL_MSG_BUFFER_SIZE             4096
#define ASL_MAX_DISABLED_MESSAGES       32
#define HEX_TABLE_LINE_SIZE             8
#define HEX_LISTING_LINE_SIZE           8


/* Source code buffers and pointers for error reporting */

ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_CurrentLineBuffer, NUL
ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_LineBufPtr, NULL);
ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_LineBufferSize, ASL_DEF
ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentColumn, 0);
ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_PreviousLineNumber, 0);
ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentLineNumber, 1);
```

```
126 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_LogicalLineNumber, 1);
127 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentLineOffset, 0);

129 /* Exception reporting */

131 ASL_EXTERN ASL_ERROR_MSG            ASL_INIT_GLOBAL (*Gbl_ErrorLog,NULL);
132 ASL_EXTERN ASL_ERROR_MSG            ASL_INIT_GLOBAL (*Gbl_NextError,NULL);

134 /* Option flags */

136 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DoCompile, TRUE);
137 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DoSignon, TRUE);
138 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_PreprocessOnly, FALSE);
139 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_PreprocessFlag, TRUE);
140 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DisassembleAll, FALSE);

142 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_UseDefaultAmlFilename,
143 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_NsOutputFlag, FALSE);
144 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_PreprocessorOutputFlag,
145 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DebugFlag, FALSE);
146 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_AsmOutputFlag, FALSE);
147 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_C_OutputFlag, FALSE);
148 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_C_OffsetTableFlag, FALS
149 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_AsmIncludeOutputFlag, F
150 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_C_IncludeOutputFlag, FA
151 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_ListingFlag, FALSE);
152 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_IgnoreErrors, FALSE);
153 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_SourceOutputFlag, FALSE
154 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_ParseOnlyFlag, FALSE);
155 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_CompileTimesFlag, FALSE
156 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_FoldConstants, TRUE);
157 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_VerboseErrors, TRUE);
158 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_NoErrors, FALSE);
159 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_WarningsAsErrors, FALSE
160 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_NoResourceChecking, FAL
161 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DisasmFlag, FALSE);
162 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_GetAllTables, FALSE);
163 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_IntegerOptimizationFlag
164 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_ReferenceOptimizationFl
165 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DisplayRemarks, TRUE);
166 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DisplayWarnings, TRUE);
167 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DisplayOptimizations, F
168 ASL_EXTERN UINT8                    ASL_INIT_GLOBAL (Gbl_WarningLevel, ASL_WARNI
169 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_UseOriginalCompilerId,
170 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_VerboseTemplates, FALSE
171 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_DoTemplates, FALSE);
172 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_CompileGeneric, FALSE);


175 #define HEX_OUTPUT_NONE             0
176 #define HEX_OUTPUT_C                1
177 #define HEX_OUTPUT_ASM              2
178 #define HEX_OUTPUT_ASL              3

180 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_HexOutputFlag, HEX_OUTP


183 /* Files */

185 ASL_EXTERN char                     *Gbl_DirectoryPath;
186 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_IncludeFilename, NULL)
187 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_OutputFilenamePrefix,
188 ASL_EXTERN ASL_INCLUDE_DIR          ASL_INIT_GLOBAL (*Gbl_IncludeDirList, NULL);
189 ASL_EXTERN char                     *Gbl_CurrentInputFilename;
190 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_ExternalRefFilename, N
```



```
192 ASL_EXTERN BOOLEAN                  ASL_INIT_GLOBAL (Gbl_HasIncludeFiles, FALSE)


195 /* Statistics */

197 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_InputByteCount, 0);
198 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_InputFieldCount, 0);
199 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_NsLookupCount, 0);
200 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalKeywords, 0);
201 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalNamedObjects, 0);
202 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalExecutableOpcodes, 0);
203 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalParseNodes, 0);
204 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalMethods, 0);
205 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalAllocations, 0);
206 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalAllocated, 0);
207 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (TotalFolds, 0);


210 /* Misc */

212 ASL_EXTERN UINT8                    ASL_INIT_GLOBAL (Gbl_RevisionOverride, 0);
213 ASL_EXTERN UINT8                    ASL_INIT_GLOBAL (Gbl_TempCount, 0);
214 ASL_EXTERN ACPI_PARSE_OBJECT        ASL_INIT_GLOBAL (*RootNode, NULL);
215 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_TableLength, 0);
216 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_SourceLine, 0);
217 ASL_EXTERN ASL_LISTING_NODE         ASL_INIT_GLOBAL (*Gbl_ListingNode, NULL);
218 ASL_EXTERN ACPI_PARSE_OBJECT        ASL_INIT_GLOBAL (*Gbl_NodeCacheNext, NULL);
219 ASL_EXTERN ACPI_PARSE_OBJECT        ASL_INIT_GLOBAL (*Gbl_NodeCacheLast, NULL);
220 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_StringCacheNext, NULL)
221 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_StringCacheLast, NULL)
222 ASL_EXTERN ACPI_PARSE_OBJECT        *Gbl_FirstLevelInsertionNode;
223 ASL_EXTERN UINT8                    ASL_INIT_GLOBAL (Gbl_FileType, 0);
224 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_Signature, NULL);
225 ASL_EXTERN char                     *Gbl_TemplateSignature;

227 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentHexColumn, 0);
228 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentAmlOffset, 0);
229 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_CurrentLine, 0);
230 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_DisabledMessagesIndex,
231 ASL_EXTERN UINT8                    ASL_INIT_GLOBAL (Gbl_HexBytesWereWritten, FA
232 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_NumNamespaceObjects, 0)
233 ASL_EXTERN UINT32                   ASL_INIT_GLOBAL (Gbl_ReservedMethods, 0);
234 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_TableSignature, "NO_SI
235 ASL_EXTERN char                     ASL_INIT_GLOBAL (*Gbl_TableId, "NO_ID");


238 /* Static structures */

240 ASL_EXTERN ASL_ANALYSIS_WALK_INFO   AnalysisWalkInfo;
241 ASL_EXTERN ACPI_TABLE_HEADER        TableHeader;

243 /* Event timing */

245 #define ASL_NUM_EVENTS              20
246 ASL_EXTERN ASL_EVENT_INFO           AslGbl_Events[ASL_NUM_EVENTS];
247 ASL_EXTERN UINT8                    AslGbl_NextEvent;
248 ASL_EXTERN UINT8                    AslGbl_NamespaceEvent;

250 /* Scratch buffers */

252 ASL_EXTERN UINT8                    Gbl_AmlBuffer[HEX_LISTING_LINE_SIZE];
253 ASL_EXTERN char                     MsgBuffer[ASL_MSG_BUFFER_SIZE];
254 ASL_EXTERN char                     StringBuffer[ASL_MSG_BUFFER_SIZE];
255 ASL_EXTERN char                     StringBuffer2[ASL_MSG_BUFFER_SIZE];
256 ASL_EXTERN UINT32                   Gbl_DisabledMessages[ASL_MAX_DISABLED_MESSAG
```

```
259 #endif /* __ASLGLOBAL_H */
```

```
**********************************************************
   11549 Thu Dec 26 13:48:28 2013
new/usr/src/common/acpica/compiler/aslhex.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslhex - ASCII hex output file generation (C, ASM, and ASL)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("ashex")

/*
 * This module emits ASCII hex output files in either C, ASM, or ASL format
 */


/* Local prototypes */

static void
HxDoHexOutputC (
    void);

static void
```



```
HxDoHexOutputAsl (
    void);

static void
HxDoHexOutputAsm (
    void);

static UINT32
HxReadAmlOutputFile (
    UINT8                   *Buffer);


/******************************************************************************
 *
 * FUNCTION:    HxDoHexOutput
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Create the hex output file. Note: data is obtained by reading
 *              the entire AML output file that was previously generated.
 *
 *****************************************************************************/

void
HxDoHexOutput (
    void)
{

    switch (Gbl_HexOutputFlag)
    {
    case HEX_OUTPUT_C:

        HxDoHexOutputC ();
        break;

    case HEX_OUTPUT_ASM:

        HxDoHexOutputAsm ();
        break;

    case HEX_OUTPUT_ASL:

        HxDoHexOutputAsl ();
        break;

    default:

        /* No other output types supported */

        break;
    }
}


/******************************************************************************
 *
 * FUNCTION:    HxReadAmlOutputFile
 *
 * PARAMETERS:  Buffer              - Where to return data
 *
 * RETURN:      None
 *
 * DESCRIPTION: Read a line of the AML output prior to formatting the data
 *
```

```
 *****************************************************************************/

static UINT32
HxReadAmlOutputFile (
    UINT8                   *Buffer)
{
    UINT32                  Actual;


    Actual = fread (Buffer, 1, HEX_TABLE_LINE_SIZE,
        Gbl_Files[ASL_FILE_AML_OUTPUT].Handle);

    if (ferror (Gbl_Files[ASL_FILE_AML_OUTPUT].Handle))
    {
        FlFileError (ASL_FILE_AML_OUTPUT, ASL_MSG_READ);
        AslAbort ();
    }

    return (Actual);
}


/*****************************************************************************
 *
 * FUNCTION:    HxDoHexOutputC
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Create the hex output file. This is the same data as the AML
 *              output file, but formatted into hex/ascii bytes suitable for
 *              inclusion into a C source file.
 *
 *****************************************************************************/

static void
HxDoHexOutputC (
    void)
{
    UINT8                   FileData[HEX_TABLE_LINE_SIZE];
    UINT32                  LineLength;
    UINT32                  Offset = 0;
    UINT32                  AmlFileSize;
    UINT32                  i;


    /* Get AML size, seek back to start */

    AmlFileSize = FlGetFileSize (ASL_FILE_AML_OUTPUT);
    FlSeekFile (ASL_FILE_AML_OUTPUT, 0);

    FlPrintFile (ASL_FILE_HEX_OUTPUT, " * C source code output\n");
    FlPrintFile (ASL_FILE_HEX_OUTPUT, " * AML code block contains 0x%X bytes\n *
        AmlFileSize);
    FlPrintFile (ASL_FILE_HEX_OUTPUT, "unsigned char AmlCode[] =\n{\n");

    while (Offset < AmlFileSize)
    {
        /* Read enough bytes needed for one output line */

        LineLength = HxReadAmlOutputFile (FileData);
        if (!LineLength)
        {
            break;
        }
```



```
        FlPrintFile (ASL_FILE_HEX_OUTPUT, "    ");

        for (i = 0; i < LineLength; i++)
        {
            /*
             * Print each hex byte.
             * Add a comma until the very last byte of the AML file
             * (Some C compilers complain about a trailing comma)
             */
            FlPrintFile (ASL_FILE_HEX_OUTPUT, "0x%2.2X", FileData[i]);
            if ((Offset + i + 1) < AmlFileSize)
            {
                FlPrintFile (ASL_FILE_HEX_OUTPUT, ",");
            }
            else
            {
                FlPrintFile (ASL_FILE_HEX_OUTPUT, " ");
            }
        }

        /* Add fill spaces if needed for last line */

        if (LineLength < HEX_TABLE_LINE_SIZE)
        {
            FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s",
                5 * (HEX_TABLE_LINE_SIZE - LineLength), " ");
        }

        /* Emit the offset and ascii dump for the entire line */

        FlPrintFile (ASL_FILE_HEX_OUTPUT, "  /* %8.8X", Offset);
        LsDumpAsciiInComment (ASL_FILE_HEX_OUTPUT, LineLength, FileData);
        FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s*/\n",
            HEX_TABLE_LINE_SIZE - LineLength + 1, " ");

        Offset += LineLength;
    }

    FlPrintFile (ASL_FILE_HEX_OUTPUT, "};\n");
}


/*****************************************************************************
 *
 * FUNCTION:    HxDoHexOutputAsl
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Create the hex output file. This is the same data as the AML
 *              output file, but formatted into hex/ascii bytes suitable for
 *              inclusion into a C source file.
 *
 *****************************************************************************/

static void
HxDoHexOutputAsl (
    void)
{
    UINT8                   FileData[HEX_TABLE_LINE_SIZE];
    UINT32                  LineLength;
    UINT32                  Offset = 0;
    UINT32                  AmlFileSize;
    UINT32                  i;
```

```
262     /* Get AML size, seek back to start */
264     AmlFileSize = FlGetFileSize (ASL_FILE_AML_OUTPUT);
265     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
267     FlPrintFile (ASL_FILE_HEX_OUTPUT, " * ASL source code output\n");
268     FlPrintFile (ASL_FILE_HEX_OUTPUT, " * AML code block contains 0x%X bytes\n *
269         AmlFileSize);
270     FlPrintFile (ASL_FILE_HEX_OUTPUT, "    Name (BUF1, Buffer()\n    {\n");
272     while (Offset < AmlFileSize)
273     {
274         /* Read enough bytes needed for one output line */
276         LineLength = HxReadAmlOutputFile (FileData);
277         if (!LineLength)
278         {
279             break;
280         }
282         FlPrintFile (ASL_FILE_HEX_OUTPUT, "        ");
284         for (i = 0; i < LineLength; i++)
285         {
286             /*
287              * Print each hex byte.
288              * Add a comma until the very last byte of the AML file
289              * (Some C compilers complain about a trailing comma)
290              */
291             FlPrintFile (ASL_FILE_HEX_OUTPUT, "0x%2.2X", FileData[i]);
292             if ((Offset + i + 1) < AmlFileSize)
293             {
294                 FlPrintFile (ASL_FILE_HEX_OUTPUT, ",");
295             }
296             else
297             {
298                 FlPrintFile (ASL_FILE_HEX_OUTPUT, " ");
299             }
300         }
302         /* Add fill spaces if needed for last line */
304         if (LineLength < HEX_TABLE_LINE_SIZE)
305         {
306             FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s",
307                 5 * (HEX_TABLE_LINE_SIZE - LineLength), " ");
308         }
310         /* Emit the offset and ascii dump for the entire line */
312         FlPrintFile (ASL_FILE_HEX_OUTPUT, "  /* %8.8X", Offset);
313         LsDumpAsciiInComment (ASL_FILE_HEX_OUTPUT, LineLength, FileData);
314         FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s*/\n",
315             HEX_TABLE_LINE_SIZE - LineLength + 1, " ");
317         Offset += LineLength;
318     }
320     FlPrintFile (ASL_FILE_HEX_OUTPUT, "    })\n");
321 }
324 /******************************************************************************
325  *
```



```
326  * FUNCTION:    HxDoHexOutputAsm
327  *
328  * PARAMETERS:  None
329  *
330  * RETURN:      None
331  *
332  * DESCRIPTION: Create the hex output file. This is the same data as the AML
333  *              output file, but formatted into hex/ascii bytes suitable for
334  *              inclusion into a ASM source file.
335  *
336  *****************************************************************************/
338 static void
339 HxDoHexOutputAsm (
340     void)
341 {
342     UINT8                   FileData[HEX_TABLE_LINE_SIZE];
343     UINT32                  LineLength;
344     UINT32                  Offset = 0;
345     UINT32                  AmlFileSize;
346     UINT32                  i;
349     /* Get AML size, seek back to start */
351     AmlFileSize = FlGetFileSize (ASL_FILE_AML_OUTPUT);
352     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
354     FlPrintFile (ASL_FILE_HEX_OUTPUT, "; Assembly code source output\n");
355     FlPrintFile (ASL_FILE_HEX_OUTPUT, "; AML code block contains 0x%X bytes\n;\n
356         AmlFileSize);
358     while (Offset < AmlFileSize)
359     {
360         /* Read enough bytes needed for one output line */
362         LineLength = HxReadAmlOutputFile (FileData);
363         if (!LineLength)
364         {
365             break;
366         }
368         FlPrintFile (ASL_FILE_HEX_OUTPUT, "  db  ");
370         for (i = 0; i < LineLength; i++)
371         {
372             /*
373              * Print each hex byte.
374              * Add a comma until the last byte of the line
375              */
376             FlPrintFile (ASL_FILE_HEX_OUTPUT, "0%2.2Xh", FileData[i]);
377             if ((i + 1) < LineLength)
378             {
379                 FlPrintFile (ASL_FILE_HEX_OUTPUT, ",");
380             }
381         }
383         FlPrintFile (ASL_FILE_HEX_OUTPUT, " ");
385         /* Add fill spaces if needed for last line */
387         if (LineLength < HEX_TABLE_LINE_SIZE)
388         {
389             FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s",
390                 5 * (HEX_TABLE_LINE_SIZE - LineLength), " ");
391         }
```

```
393         /* Emit the offset and ascii dump for the entire line */

395         FlPrintFile (ASL_FILE_HEX_OUTPUT, "  ; %8.8X", Offset);
396         LsDumpAsciiInComment (ASL_FILE_HEX_OUTPUT, LineLength, FileData);
397         FlPrintFile (ASL_FILE_HEX_OUTPUT, "\n");

399         Offset += LineLength;
400     }

402     FlPrintFile (ASL_FILE_HEX_OUTPUT, "\n");
403 }
```

```
**********************************************************
   12065 Thu Dec 26 13:48:28 2013
new/usr/src/common/acpica/compiler/asllength.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asllength - Tree walk to determine package and opcode lengths
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asllength")

/* Local prototypes */

static UINT8
CgGetPackageLenByteCount (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  PackageLength);

static void
```



```
CgGenerateAmlOpcodeLength (
    ACPI_PARSE_OBJECT       *Op);


#ifdef ACPI_OBSOLETE_FUNCTIONS
void
LnAdjustLengthToRoot (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  LengthDelta);
#endif


/*******************************************************************************
 *
 * FUNCTION:    LnInitLengthsWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk callback to initialize (and re-initialize) the node
 *              subtree length(s) to zero. The Subtree lengths are bubbled
 *              up to the root node in order to get a total AML length.
 *
 ******************************************************************************/

ACPI_STATUS
LnInitLengthsWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{

    Op->Asl.AmlSubtreeLength = 0;
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    LnPackageLengthWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk callback to calculate the total AML length.
 *              1) Calculate the AML lengths (opcode, package length, etc.) for
 *                 THIS node.
 *              2) Bubbble up all of these lengths to the parent node by summing
 *                 them all into the parent subtree length.
 *
 * Note:  The SubtreeLength represents the total AML length of all child nodes
 *        in all subtrees under a given node. Therefore, once this walk is
 *        complete, the Root Node subtree length is the AML length of the entire
 *        tree (and thus, the entire ACPI table)
 *
 ******************************************************************************/

ACPI_STATUS
LnPackageLengthWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
```

```
127     /* Generate the AML lengths for this node */

129     CgGenerateAmlLengths (Op);

131     /* Bubble up all lengths (this node and all below it) to the parent */

133     if ((Op->Asl.Parent) &&
134         (Op->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG))
135     {
136         Op->Asl.Parent->Asl.AmlSubtreeLength += (Op->Asl.AmlLength +
137                                                 Op->Asl.AmlOpcodeLength +
138                                                 Op->Asl.AmlPkgLenBytes +
139                                                 Op->Asl.AmlSubtreeLength);
140     }
141     return (AE_OK);
142 }


145 /*******************************************************************************
146  *
147  * FUNCTION:    CgGetPackageLenByteCount
148  *
149  * PARAMETERS:  Op              - Parse node
150  *              PackageLength   - Length to be encoded
151  *
152  * RETURN:      Required length of the package length encoding
153  *
154  * DESCRIPTION: Calculate the number of bytes required to encode the given
155  *              package length.
156  *
157  ******************************************************************************/

159 static UINT8
160 CgGetPackageLenByteCount (
161     ACPI_PARSE_OBJECT       *Op,
162     UINT32                  PackageLength)
163 {

165     /*
166      * Determine the number of bytes required to encode the package length
167      * Note: the package length includes the number of bytes used to encode
168      * the package length, so we must account for this also.
169      */
170     if (PackageLength <= (0x0000003F - 1))
171     {
172         return (1);
173     }
174     else if (PackageLength <= (0x00000FFF - 2))
175     {
176         return (2);
177     }
178     else if (PackageLength <= (0x000FFFFF - 3))
179     {
180         return (3);
181     }
182     else if (PackageLength <= (0x0FFFFFFF - 4))
183     {
184         return (4);
185     }
186     else
187     {
188         /* Fatal error - the package length is too large to encode */

190         AslError (ASL_ERROR, ASL_MSG_ENCODING_LENGTH, Op, NULL);
191     }
```



```
193     return (0);
194 }


197 /*******************************************************************************
198  *
199  * FUNCTION:    CgGenerateAmlOpcodeLength
200  *
201  * PARAMETERS:  Op          - Parse node whose AML opcode lengths will be
202  *                            calculated
203  *
204  * RETURN:      None.
205  *
206  * DESCRIPTION: Calculate the AmlOpcodeLength, AmlPkgLenBytes, and AmlLength
207  *              fields for this node.
208  *
209  ******************************************************************************/

211 static void
212 CgGenerateAmlOpcodeLength (
213     ACPI_PARSE_OBJECT       *Op)
214 {

216     /* Check for two-byte opcode */

218     if (Op->Asl.AmlOpcode > 0x00FF)
219     {
220         Op->Asl.AmlOpcodeLength = 2;
221     }
222     else
223     {
224         Op->Asl.AmlOpcodeLength = 1;
225     }

227     /* Does this opcode have an associated "PackageLength" field? */

229     Op->Asl.AmlPkgLenBytes = 0;
230     if (Op->Asl.CompileFlags & NODE_AML_PACKAGE)
231     {
232         Op->Asl.AmlPkgLenBytes = CgGetPackageLenByteCount (
233                                     Op, Op->Asl.AmlSubtreeLength);
234     }

236     /* Data opcode lengths are easy */

238     switch (Op->Asl.AmlOpcode)
239     {
240     case AML_BYTE_OP:

242         Op->Asl.AmlLength = 1;
243         break;

245     case AML_WORD_OP:

247         Op->Asl.AmlLength = 2;
248         break;

250     case AML_DWORD_OP:

252         Op->Asl.AmlLength = 4;
253         break;

255     case AML_QWORD_OP:

257         Op->Asl.AmlLength = 8;
258         break;
```

```
260     default:

262         /* All data opcodes must be above */
263         break;
264     }
265 }


268 /*******************************************************************************
269  *
270  * FUNCTION:    CgGenerateAmlLengths
271  *
272  * PARAMETERS:  Op        - Parse node
273  *
274  * RETURN:      None.
275  *
276  * DESCRIPTION: Generate internal length fields based on the AML opcode or
277  *              parse opcode.
278  *
279  ******************************************************************************/

281 void
282 CgGenerateAmlLengths (
283     ACPI_PARSE_OBJECT       *Op)
284 {
285     char                    *Buffer;
286     ACPI_STATUS             Status;


289     switch (Op->Asl.AmlOpcode)
290     {
291     case AML_RAW_DATA_BYTE:

293         Op->Asl.AmlOpcodeLength = 0;
294         Op->Asl.AmlLength = 1;
295         return;

297     case AML_RAW_DATA_WORD:

299         Op->Asl.AmlOpcodeLength = 0;
300         Op->Asl.AmlLength = 2;
301         return;

303     case AML_RAW_DATA_DWORD:

305         Op->Asl.AmlOpcodeLength = 0;
306         Op->Asl.AmlLength = 4;
307         return;

309     case AML_RAW_DATA_QWORD:

311         Op->Asl.AmlOpcodeLength = 0;
312         Op->Asl.AmlLength = 8;
313         return;

315     case AML_RAW_DATA_BUFFER:

317         /* Aml length is/was set by creator */

319         Op->Asl.AmlOpcodeLength = 0;
320         return;

322     case AML_RAW_DATA_CHAIN:

324         /* Aml length is/was set by creator */
```



```
326         Op->Asl.AmlOpcodeLength = 0;
327         return;

329     default:

331         break;
332     }

334     switch (Op->Asl.ParseOpcode)
335     {
336     case PARSEOP_DEFINITIONBLOCK:

338         Gbl_TableLength = sizeof (ACPI_TABLE_HEADER) +
339                           Op->Asl.AmlSubtreeLength;
340         break;

342     case PARSEOP_NAMESEG:

344         Op->Asl.AmlOpcodeLength = 0;
345         Op->Asl.AmlLength = 4;
346         Op->Asl.ExternalName = Op->Asl.Value.String;
347         break;

349     case PARSEOP_NAMESTRING:
350     case PARSEOP_METHODCALL:

352         if (Op->Asl.CompileFlags & NODE_NAME_INTERNALIZED)
353         {
354             break;
355         }

357         Op->Asl.AmlOpcodeLength = 0;
358         Status = UtInternalizeName (Op->Asl.Value.String, &Buffer);
359         if (ACPI_FAILURE (Status))
360         {
361             DbgPrint (ASL_DEBUG_OUTPUT,
362                 "Failure from internalize name %X\n", Status);
363             break;
364         }

366         Op->Asl.ExternalName = Op->Asl.Value.String;
367         Op->Asl.Value.String = Buffer;
368         Op->Asl.CompileFlags |= NODE_NAME_INTERNALIZED;

370         Op->Asl.AmlLength = strlen (Buffer);

372         /*
373          * Check for single backslash reference to root,
374          * make it a null terminated string in the AML
375          */
376         if (Op->Asl.AmlLength == 1)
377         {
378             Op->Asl.AmlLength = 2;
379         }
380         break;

382     case PARSEOP_STRING_LITERAL:

384         Op->Asl.AmlOpcodeLength = 1;

386         /* Get null terminator */

388         Op->Asl.AmlLength = strlen (Op->Asl.Value.String) + 1;
389         break;
```

```
391     case PARSEOP_PACKAGE_LENGTH:

393         Op->Asl.AmlOpcodeLength = 0;
394         Op->Asl.AmlPkgLenBytes = CgGetPackageLenByteCount (Op,
395                                     (UINT32) Op->Asl.Value.Integer);
396         break;

398     case PARSEOP_RAW_DATA:

400         Op->Asl.AmlOpcodeLength = 0;
401         break;

403     case PARSEOP_DEFAULT_ARG:
404     case PARSEOP_EXTERNAL:
405     case PARSEOP_INCLUDE:
406     case PARSEOP_INCLUDE_END:

408         /* Ignore the "default arg" nodes, they are extraneous at this point */

410         break;

412     default:

414         CgGenerateAmlOpcodeLength (Op);
415         break;
416     }
417 }


420 #ifdef ACPI_OBSOLETE_FUNCTIONS
421 /*******************************************************************************
422  *
423  * FUNCTION:    LnAdjustLengthToRoot
424  *
425  * PARAMETERS:  Op      - Node whose Length was changed
426  *
427  * RETURN:      None.
428  *
429  * DESCRIPTION: Change the Subtree length of the given node, and bubble the
430  *              change all the way up to the root node. This allows for
431  *              last second changes to a package length (for example, if the
432  *              package length encoding gets shorter or longer.)
433  *
434  ******************************************************************************/

436 void
437 LnAdjustLengthToRoot (
438     ACPI_PARSE_OBJECT       *SubtreeOp,
439     UINT32                  LengthDelta)
440 {
441     ACPI_PARSE_OBJECT       *Op;


444     /* Adjust all subtree lengths up to the root */

446     Op = SubtreeOp->Asl.Parent;
447     while (Op)
448     {
449         Op->Asl.AmlSubtreeLength -= LengthDelta;
450         Op = Op->Asl.Parent;
451     }

453     /* Adjust the global table length */

455     Gbl_TableLength -= LengthDelta;
456 }
```



```
457 #endif
```

```
**********************************************************
   17879 Thu Dec 26 13:48:29 2013
new/usr/src/common/acpica/compiler/asllisting.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asllisting - Listing file generation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"
#include "acparser.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asllisting")


/* Local prototypes */

static void
LsGenerateListing (
    UINT32                  FileId);
```



```
static ACPI_STATUS
LsAmlListingWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
LsTreeWriteWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static void
LsWriteNodeToListing (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  FileId);

static void
LsFinishSourceListing (
    UINT32                  FileId);


/*******************************************************************************
 *
 * FUNCTION:    LsDoListings
 *
 * PARAMETERS:  None. Examines the various output file global flags.
 *
 * RETURN:      None
 *
 * DESCRIPTION: Generate all requested listing files.
 *
 ******************************************************************************/

void
LsDoListings (
    void)
{

    if (Gbl_C_OutputFlag)
    {
        LsGenerateListing (ASL_FILE_C_SOURCE_OUTPUT);
    }

    if (Gbl_ListingFlag)
    {
        LsGenerateListing (ASL_FILE_LISTING_OUTPUT);
    }

    if (Gbl_AsmOutputFlag)
    {
        LsGenerateListing (ASL_FILE_ASM_SOURCE_OUTPUT);
    }

    if (Gbl_C_IncludeOutputFlag)
    {
        LsGenerateListing (ASL_FILE_C_INCLUDE_OUTPUT);
    }

    if (Gbl_AsmIncludeOutputFlag)
    {
        LsGenerateListing (ASL_FILE_ASM_INCLUDE_OUTPUT);
    }

    if (Gbl_C_OffsetTableFlag)
    {
```

```
127         LsGenerateListing (ASL_FILE_C_OFFSET_OUTPUT);
128     }
129 }


132 /*******************************************************************************
133  *
134  * FUNCTION:    LsGenerateListing
135  *
136  * PARAMETERS:  FileId      - ID of listing file
137  *
138  * RETURN:      None
139  *
140  * DESCRIPTION: Generate a listing file. This can be one of the several types
141  *              of "listings" supported.
142  *
143  ******************************************************************************/

145 static void
146 LsGenerateListing (
147     UINT32                  FileId)
148 {

150     /* Start at the beginning of both the source and AML files */

152     FlSeekFile (ASL_FILE_SOURCE_OUTPUT, 0);
153     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
154     Gbl_SourceLine = 0;
155     Gbl_CurrentHexColumn = 0;
156     LsPushNode (Gbl_Files[ASL_FILE_INPUT].Filename);

158     if (FileId == ASL_FILE_C_OFFSET_OUTPUT)
159     {
160         Gbl_CurrentAmlOffset = 0;

162         /* Offset table file has a special header and footer */

164         LsDoOffsetTableHeader (FileId);

166         TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD, LsAmlOffsetWalk,
167             NULL, (void *) ACPI_TO_POINTER (FileId));
168         LsDoOffsetTableFooter (FileId);
169         return;
170     }

172     /* Process all parse nodes */

174     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD, LsAmlListingWalk,
175         NULL, (void *) ACPI_TO_POINTER (FileId));

177     /* Final processing */

179     LsFinishSourceListing (FileId);
180 }


183 /*******************************************************************************
184  *
185  * FUNCTION:    LsAmlListingWalk
186  *
187  * PARAMETERS:  ASL_WALK_CALLBACK
188  *
189  * RETURN:      Status
190  *
191  * DESCRIPTION: Process one node during a listing file generation.
192  *
```



```
193  ******************************************************************************/

195 static ACPI_STATUS
196 LsAmlListingWalk (
197     ACPI_PARSE_OBJECT       *Op,
198     UINT32                  Level,
199     void                    *Context)
200 {
201     UINT8                   FileByte;
202     UINT32                  i;
203     UINT32                  FileId = (UINT32) ACPI_TO_INTEGER (Context);


206     LsWriteNodeToListing (Op, FileId);

208     if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DATA)
209     {
210         /* Buffer is a resource template, don't dump the data all at once */

212         return (AE_OK);
213     }

215     /* Write the hex bytes to the listing file(s) (if requested) */

217     for (i = 0; i < Op->Asl.FinalAmlLength; i++)
218     {
219         if (ACPI_FAILURE (FlReadFile (ASL_FILE_AML_OUTPUT, &FileByte, 1)))
220         {
221             FlFileError (ASL_FILE_AML_OUTPUT, ASL_MSG_READ);
222             AslAbort ();
223         }
224         LsWriteListingHexBytes (&FileByte, 1, FileId);
225     }

227     return (AE_OK);
228 }


231 /*******************************************************************************
232  *
233  * FUNCTION:    LsDumpParseTree, LsTreeWriteWalk
234  *
235  * PARAMETERS:  None
236  *
237  * RETURN:      None
238  *
239  * DESCRIPTION: Dump entire parse tree, for compiler debug only
240  *
241  ******************************************************************************/

243 void
244 LsDumpParseTree (
245     void)
246 {

248     if (!Gbl_DebugFlag)
249     {
250         return;
251     }

253     DbgPrint (ASL_TREE_OUTPUT, "\nOriginal parse tree from parser:\n\n");
254     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
255         LsTreeWriteWalk, NULL, NULL);
256 }
```

```
259 static ACPI_STATUS
260 LsTreeWriteWalk (
261     ACPI_PARSE_OBJECT       *Op,
262     UINT32                  Level,
263     void                    *Context)
264 {

266     /* Debug output */

268     DbgPrint (ASL_TREE_OUTPUT,
269         "%5.5d [%2d]", Op->Asl.LogicalLineNumber, Level);

271     UtPrintFormattedName (Op->Asl.ParseOpcode, Level);

273     DbgPrint (ASL_TREE_OUTPUT, "    (%.4X)\n", Op->Asl.ParseOpcode);
274     return (AE_OK);
275 }


278 /*******************************************************************************
279  *
280  * FUNCTION:    LsWriteNodeToListing
281  *
282  * PARAMETERS:  Op              - Parse node to write to the listing file.
283  *              FileId          - ID of current listing file
284  *
285  * RETURN:      None.
286  *
287  * DESCRIPTION: Write "a node" to the listing file. This means to
288  *              1) Write out all of the source text associated with the node
289  *              2) Write out all of the AML bytes associated with the node
290  *              3) Write any compiler exceptions associated with the node
291  *
292  ******************************************************************************/

294 static void
295 LsWriteNodeToListing (
296     ACPI_PARSE_OBJECT       *Op,
297     UINT32                  FileId)
298 {
299     const ACPI_OPCODE_INFO  *OpInfo;
300     UINT32                  OpClass;
301     char                    *Pathname;
302     UINT32                  Length;
303     UINT32                  i;


306     OpInfo  = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
307     OpClass = OpInfo->Class;

309     /* TBD: clean this up with a single flag that says:
310      * I start a named output block
311      */
312     if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
313     {
314         switch (Op->Asl.ParseOpcode)
315         {
316         case PARSEOP_DEFINITIONBLOCK:
317         case PARSEOP_METHODCALL:
318         case PARSEOP_INCLUDE:
319         case PARSEOP_INCLUDE_END:
320         case PARSEOP_DEFAULT_ARG:

322             break;

324         default:
```



```
326             switch (OpClass)
327             {
328             case AML_CLASS_NAMED_OBJECT:

330                 switch (Op->Asl.AmlOpcode)
331                 {
332                 case AML_SCOPE_OP:
333                 case AML_ALIAS_OP:

335                     break;

337                 default:

339                     if (Op->Asl.ExternalName)
340                     {
341                         LsFlushListingBuffer (FileId);
342                         FlPrintFile (FileId, "    };\n");
343                     }
344                     break;
345                 }
346                 break;

348             default:

350                 /* Don't care about other objects */

352                 break;
353             }
354             break;
355         }
356     }

358     /* These cases do not have a corresponding AML opcode */

360     switch (Op->Asl.ParseOpcode)
361     {
362     case PARSEOP_DEFINITIONBLOCK:

364         LsWriteSourceLines (Op->Asl.EndLine, Op->Asl.EndLogicalLine, FileId);

366         /* Use the table Signature and TableId to build a unique name */

368         if (FileId == ASL_FILE_ASM_SOURCE_OUTPUT)
369         {
370             FlPrintFile (FileId,
371                 "%s_%s_Header \\\n",
372                 Gbl_TableSignature, Gbl_TableId);
373         }
374         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
375         {
376             FlPrintFile (FileId,
377                 "    unsigned char    %s_%s_Header [] =\n    {\n",
378                 Gbl_TableSignature, Gbl_TableId);
379         }
380         if (FileId == ASL_FILE_ASM_INCLUDE_OUTPUT)
381         {
382             FlPrintFile (FileId,
383                 "extrn %s_%s_Header : byte\n",
384                 Gbl_TableSignature, Gbl_TableId);
385         }
386         if (FileId == ASL_FILE_C_INCLUDE_OUTPUT)
387         {
388             FlPrintFile (FileId,
389                 "extern unsigned char    %s_%s_Header [];\n",
390                 Gbl_TableSignature, Gbl_TableId);
```

```
391         }
392         return;


395     case PARSEOP_METHODCALL:

397         LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumber,
398             FileId);
399         return;


402     case PARSEOP_INCLUDE:

404         /* Flush everything up to and including the include source line */

406         LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumber,
407             FileId);

409         /* Create a new listing node and push it */

411         LsPushNode (Op->Asl.Child->Asl.Value.String);
412         return;


415     case PARSEOP_INCLUDE_END:

417         /* Flush out the rest of the include file */

419         LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumber,
420             FileId);

422         /* Pop off this listing node and go back to the parent file */

424         (void) LsPopNode ();
425         return;


428     case PARSEOP_DEFAULT_ARG:

430         if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC)
431         {
432             LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.EndLogicalLine,
433                 FileId);
434         }
435         return;


438     default:

440         /* All other opcodes have an AML opcode */

442         break;
443     }

445     /*
446      * Otherwise, we look at the AML opcode because we can
447      * switch on the opcode type, getting an entire class
448      * at once
449      */
450     switch (OpClass)
451     {
452     case AML_CLASS_ARGUMENT:       /* argument type only */
453     case AML_CLASS_INTERNAL:

455         break;
```



```
457     case AML_CLASS_NAMED_OBJECT:

459         switch (Op->Asl.AmlOpcode)
460         {
461         case AML_FIELD_OP:
462         case AML_INDEX_FIELD_OP:
463         case AML_BANK_FIELD_OP:
464             /*
465              * For fields, we want to dump all the AML after the
466              * entire definition
467              */
468             LsWriteSourceLines (Op->Asl.EndLine, Op->Asl.EndLogicalLine,
469                 FileId);
470             break;

472         case AML_NAME_OP:

474             if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC)
475             {
476                 LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumbe
477                     FileId);
478             }
479             else
480             {
481                 /*
482                  * For fields, we want to dump all the AML after the
483                  * entire definition
484                  */
485                 LsWriteSourceLines (Op->Asl.EndLine, Op->Asl.EndLogicalLine,
486                     FileId);
487             }
488             break;

490         default:

492             LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumber,
493                 FileId);
494             break;
495         }

497         switch (Op->Asl.AmlOpcode)
498         {
499         case AML_SCOPE_OP:
500         case AML_ALIAS_OP:

502             /* These opcodes do not declare a new object, ignore them */

504             break;

506         default:

508             /* All other named object opcodes come here */

510             switch (FileId)
511             {
512             case ASL_FILE_ASM_SOURCE_OUTPUT:
513             case ASL_FILE_C_SOURCE_OUTPUT:
514             case ASL_FILE_ASM_INCLUDE_OUTPUT:
515             case ASL_FILE_C_INCLUDE_OUTPUT:
516                 /*
517                  * For named objects, we will create a valid symbol so that the
518                  * AML code can be referenced from C or ASM
519                  */
520                 if (Op->Asl.ExternalName)
521                 {
522                     /* Get the full pathname associated with this node */
```

```
524                     Pathname = AcpiNsGetExternalPathname (Op->Asl.Node);
525                     Length = strlen (Pathname);
526                     if (Length >= 4)
527                     {
528                         /* Convert all dots in the path to underscores */

530                         for (i = 0; i < Length; i++)
531                         {
532                             if (Pathname[i] == '.')
533                             {
534                                 Pathname[i] = '_';
535                             }
536                         }

538                         /* Create the appropriate symbol in the output file */

540                         if (FileId == ASL_FILE_ASM_SOURCE_OUTPUT)
541                         {
542                             FlPrintFile (FileId,
543                                 "%s_%s_%s  \\\n",
544                                 Gbl_TableSignature, Gbl_TableId, &Pathname[1]);
545                         }
546                         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
547                         {
548                             FlPrintFile (FileId,
549                                 "    unsigned char    %s_%s_%s [] =\n    {\n",
550                                 Gbl_TableSignature, Gbl_TableId, &Pathname[1]);
551                         }
552                         if (FileId == ASL_FILE_ASM_INCLUDE_OUTPUT)
553                         {
554                             FlPrintFile (FileId,
555                                 "extrn %s_%s_%s : byte\n",
556                                 Gbl_TableSignature, Gbl_TableId, &Pathname[1]);
557                         }
558                         if (FileId == ASL_FILE_C_INCLUDE_OUTPUT)
559                         {
560                             FlPrintFile (FileId,
561                                 "extern unsigned char    %s_%s_%s [];\n",
562                                 Gbl_TableSignature, Gbl_TableId, &Pathname[1]);
563                         }
564                     }
565                     ACPI_FREE (Pathname);
566                 }
567                 break;

569             default:

571                 /* Nothing to do for listing file */

573                 break;
574             }
575         }
576         break;

578     case AML_CLASS_EXECUTE:
579     case AML_CLASS_CREATE:
580     default:

582         if ((Op->Asl.ParseOpcode == PARSEOP_BUFFER) &&
583             (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC))
584         {
585             return;
586         }

588         LsWriteSourceLines (Op->Asl.LineNumber, Op->Asl.LogicalLineNumber,
```



```
589                 FileId);
590         break;

592     case AML_CLASS_UNKNOWN:

594         break;
595     }
596 }


599 /*******************************************************************************
600  *
601  * FUNCTION:    LsFinishSourceListing
602  *
603  * PARAMETERS:  FileId          - ID of current listing file.
604  *
605  * RETURN:      None
606  *
607  * DESCRIPTION: Cleanup routine for the listing file. Flush the hex AML
608  *              listing buffer, and flush out any remaining lines in the
609  *              source input file.
610  *
611  ******************************************************************************/

613 static void
614 LsFinishSourceListing (
615     UINT32                  FileId)
616 {

618     if ((FileId == ASL_FILE_ASM_INCLUDE_OUTPUT) ||
619         (FileId == ASL_FILE_C_INCLUDE_OUTPUT))
620     {
621         return;
622     }

624     LsFlushListingBuffer (FileId);
625     Gbl_CurrentAmlOffset = 0;

627     /* Flush any remaining text in the source file */

629     if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
630     {
631         FlPrintFile (FileId, "    /*\n");
632     }

634     while (LsWriteOneSourceLine (FileId))
635     { ; }

637     if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
638     {
639         FlPrintFile (FileId, "\n     */\n    };\n");
640     }

642     FlPrintFile (FileId, "\n");

644     if (FileId == ASL_FILE_LISTING_OUTPUT)
645     {
646         /* Print a summary of the compile exceptions */

648         FlPrintFile (FileId, "\n\nSummary of errors and warnings\n\n");
649         AePrintErrorLog (FileId);
650         FlPrintFile (FileId, "\n");
651         UtDisplaySummary (FileId);
652         FlPrintFile (FileId, "\n");
653     }
654 }
```

```
**********************************************************
   19439 Thu Dec 26 13:48:29 2013
new/usr/src/common/acpica/compiler/asllistsup.c
acpica-unix2-20130823
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: asllistsup - Listing file support utilities
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #include "aslcompiler.h"
  45 #include "aslcompiler.y.h"


  48 #define _COMPONENT          ACPI_COMPILER
  49         ACPI_MODULE_NAME    ("aslistsup")


  52 /*******************************************************************************
  53  *
  54  * FUNCTION:    LsDumpAscii
  55  *
  56  * PARAMETERS:  FileId          - ID of current listing file
  57  *              Count           - Number of bytes to convert
  58  *              Buffer          - Buffer of bytes to convert
  59  *
  60  * RETURN:      None
  61  *
```



```
  62  * DESCRIPTION: Convert hex bytes to ascii
  63  *
  64  ******************************************************************************/

  66 void
  67 LsDumpAscii (
  68     UINT32                  FileId,
  69     UINT32                  Count,
  70     UINT8                   *Buffer)
  71 {
  72     UINT8                   BufChar;
  73     UINT32                  i;


  76     FlPrintFile (FileId, "    \"");
  77     for (i = 0; i < Count; i++)
  78     {
  79         BufChar = Buffer[i];
  80         if (isprint (BufChar))
  81         {
  82             FlPrintFile (FileId, "%c", BufChar);
  83         }
  84         else
  85         {
  86             /* Not a printable character, just put out a dot */

  88             FlPrintFile (FileId, ".");
  89         }
  90     }
  91     FlPrintFile (FileId, "\"");
  92 }


  95 /*******************************************************************************
  96  *
  97  * FUNCTION:    LsDumpAsciiInComment
  98  *
  99  * PARAMETERS:  FileId          - ID of current listing file
 100  *              Count           - Number of bytes to convert
 101  *              Buffer          - Buffer of bytes to convert
 102  *
 103  * RETURN:      None
 104  *
 105  * DESCRIPTION: Convert hex bytes to ascii
 106  *
 107  ******************************************************************************/

 109 void
 110 LsDumpAsciiInComment (
 111     UINT32                  FileId,
 112     UINT32                  Count,
 113     UINT8                   *Buffer)
 114 {
 115     UINT8                   BufChar = 0;
 116     UINT8                   LastChar;
 117     UINT32                  i;


 120     FlPrintFile (FileId, "    \"");
 121     for (i = 0; i < Count; i++)
 122     {
 123         LastChar = BufChar;
 124         BufChar = Buffer[i];

 126         if (isprint (BufChar))
 127         {
```

```
            /* Handle embedded C comment sequences */

            if (((LastChar == '*') && (BufChar == '/')) ||
                ((LastChar == '/') && (BufChar == '*')))
            {
                /* Insert a space to break the sequence */

                FlPrintFile (FileId, ".", BufChar);
            }

            FlPrintFile (FileId, "%c", BufChar);
        }
        else
        {
            /* Not a printable character, just put out a dot */

            FlPrintFile (FileId, ".");
        }
    }

    FlPrintFile (FileId, "\"");
}


/*******************************************************************************
 *
 * FUNCTION:    LsCheckException
 *
 * PARAMETERS:  LineNumber          - Current logical (cumulative) line #
 *              FileId              - ID of output listing file
 *
 * RETURN:      None
 *
 * DESCRIPTION: Check if there is an exception for this line, and if there is,
 *              put it in the listing immediately. Handles multiple errors
 *              per line. Gbl_NextError points to the next error in the
 *              sorted (by line #) list of compile errors/warnings.
 *
 ******************************************************************************/

void
LsCheckException (
    UINT32                  LineNumber,
    UINT32                  FileId)
{

    if ((!Gbl_NextError) ||
        (LineNumber < Gbl_NextError->LogicalLineNumber ))
    {
        return;
    }

    /* Handle multiple errors per line */

    if (FileId == ASL_FILE_LISTING_OUTPUT)
    {
        while (Gbl_NextError &&
              (LineNumber >= Gbl_NextError->LogicalLineNumber))
        {
            AePrintException (FileId, Gbl_NextError, "\n[****iasl****]\n");

            Gbl_NextError = Gbl_NextError->Next;
        }

        FlPrintFile (FileId, "\n");
    }
```



```
}


/*******************************************************************************
 *
 * FUNCTION:    LsWriteListingHexBytes
 *
 * PARAMETERS:  Buffer          - AML code buffer
 *              Length          - Number of AML bytes to write
 *              FileId          - ID of current listing file.
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write the contents of the AML buffer to the listing file via
 *              the listing buffer. The listing buffer is flushed every 16
 *              AML bytes.
 *
 ******************************************************************************/

void
LsWriteListingHexBytes (
    UINT8                   *Buffer,
    UINT32                  Length,
    UINT32                  FileId)
{
    UINT32                  i;


    /* Transfer all requested bytes */

    for (i = 0; i < Length; i++)
    {
        /* Print line header when buffer is empty */

        if (Gbl_CurrentHexColumn == 0)
        {
            if (Gbl_HasIncludeFiles)
            {
                FlPrintFile (FileId, "%*s", 10, " ");
            }

            switch (FileId)
            {
            case ASL_FILE_LISTING_OUTPUT:

                FlPrintFile (FileId, "%8.8X%s", Gbl_CurrentAmlOffset,
                    ASL_LISTING_LINE_PREFIX);
                break;

            case ASL_FILE_ASM_SOURCE_OUTPUT:

                FlPrintFile (FileId, "    db ");
                break;

            case ASL_FILE_C_SOURCE_OUTPUT:

                FlPrintFile (FileId, "        ");
                break;

            default:

                /* No other types supported */

                return;
            }
        }
```

```
261         /* Transfer AML byte and update counts */

263         Gbl_AmlBuffer[Gbl_CurrentHexColumn] = Buffer[i];

265         Gbl_CurrentHexColumn++;
266         Gbl_CurrentAmlOffset++;

268         /* Flush buffer when it is full */

270         if (Gbl_CurrentHexColumn >= HEX_LISTING_LINE_SIZE)
271         {
272             LsFlushListingBuffer (FileId);
273         }
274     }
275 }


278 /******************************************************************************
279  *
280  * FUNCTION:    LsWriteSourceLines
281  *
282  * PARAMETERS:  ToLineNumber            -
283  *              ToLogicalLineNumber     - Write up to this source line number
284  *              FileId                  - ID of current listing file
285  *
286  * RETURN:      None
287  *
288  * DESCRIPTION: Read then write source lines to the listing file until we have
289  *              reached the specified logical (cumulative) line number. This
290  *              automatically echos out comment blocks and other non-AML
291  *              generating text until we get to the actual AML-generating line
292  *              of ASL code specified by the logical line number.
293  *
294  *****************************************************************************/

296 void
297 LsWriteSourceLines (
298     UINT32                  ToLineNumber,
299     UINT32                  ToLogicalLineNumber,
300     UINT32                  FileId)
301 {

303     /* Nothing to do for these file types */

305     if ((FileId == ASL_FILE_ASM_INCLUDE_OUTPUT) ||
306         (FileId == ASL_FILE_C_INCLUDE_OUTPUT))
307     {
308         return;
309     }

311     Gbl_CurrentLine = ToLogicalLineNumber;

313     /* Flush any hex bytes remaining from the last opcode */

315     LsFlushListingBuffer (FileId);

317     /* Read lines and write them as long as we are not caught up */

319     if (Gbl_SourceLine < Gbl_CurrentLine)
320     {
321         /*
322          * If we just completed writing some AML hex bytes, output a linefeed
323          * to add some whitespace for readability.
324          */
325         if (Gbl_HexBytesWereWritten)
```



```
326         {
327             FlPrintFile (FileId, "\n");
328             Gbl_HexBytesWereWritten = FALSE;
329         }

331         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
332         {
333             FlPrintFile (FileId, "    /*\n");
334         }

336         /* Write one line at a time until we have reached the target line # */

338         while ((Gbl_SourceLine < Gbl_CurrentLine) &&
339                 LsWriteOneSourceLine (FileId))
340         { ; }

342         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
343         {
344             FlPrintFile (FileId, "     */");
345         }

347         FlPrintFile (FileId, "\n");
348     }
349 }


352 /******************************************************************************
353  *
354  * FUNCTION:    LsWriteOneSourceLine
355  *
356  * PARAMETERS:  FileId          - ID of current listing file
357  *
358  * RETURN:      FALSE on EOF (input source file), TRUE otherwise
359  *
360  * DESCRIPTION: Read one line from the input source file and echo it to the
361  *              listing file, prefixed with the line number, and if the source
362  *              file contains include files, prefixed with the current filename
363  *
364  *****************************************************************************/

366 UINT32
367 LsWriteOneSourceLine (
368     UINT32                  FileId)
369 {
370     UINT8                   FileByte;
371     UINT32                  Column = 0;
372     UINT32                  Index = 16;
373     BOOLEAN                 StartOfLine = FALSE;
374     BOOLEAN                 ProcessLongLine = FALSE;


377     Gbl_SourceLine++;
378     Gbl_ListingNode->LineNumber++;

380     /* Ignore lines that are completely blank (but count the line above) */

382     if (FlReadFile (ASL_FILE_SOURCE_OUTPUT, &FileByte, 1) != AE_OK)
383     {
384         return (0);
385     }
386     if (FileByte == '\n')
387     {
388         return (1);
389     }

391     /*
```

```
392      * This is a non-empty line, we will print the entire line with
393      * the line number and possibly other prefixes and transforms.
394      */

396     /* Line prefixes for special files, C and ASM output */

398     if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
399     {
400         FlPrintFile (FileId, "     *");
401     }
402     if (FileId == ASL_FILE_ASM_SOURCE_OUTPUT)
403     {
404         FlPrintFile (FileId, "; ");
405     }

407     if (Gbl_HasIncludeFiles)
408     {
409         /*
410          * This file contains "include" statements, print the current
411          * filename and line number within the current file
412          */
413         FlPrintFile (FileId, "%12s %5d%s",
414             Gbl_ListingNode->Filename, Gbl_ListingNode->LineNumber,
415             ASL_LISTING_LINE_PREFIX);
416     }
417     else
418     {
419         /* No include files, just print the line number */

421         FlPrintFile (FileId, "%8u%s", Gbl_SourceLine,
422             ASL_LISTING_LINE_PREFIX);
423     }

425     /* Read the rest of this line (up to a newline or EOF) */

427     do
428     {
429         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
430         {
431             if (FileByte == '/')
432             {
433                 FileByte = '*';
434             }
435         }

437         /* Split long input lines for readability in the listing */

439         Column++;
440         if (Column >= 128)
441         {
442             if (!ProcessLongLine)
443             {
444                 if ((FileByte != '}') &&
445                     (FileByte != '{'))
446                 {
447                     goto WriteByte;
448                 }

450                 ProcessLongLine = TRUE;
451             }

453             if (FileByte == '{')
454             {
455                 FlPrintFile (FileId, "\n%*s{\n", Index, " ");
456                 StartOfLine = TRUE;
457                 Index += 4;
```



```
458                 continue;
459             }

461             else if (FileByte == '}')
462             {
463                 if (!StartOfLine)
464                 {
465                     FlPrintFile (FileId, "\n");
466                 }

468                 StartOfLine = TRUE;
469                 Index -= 4;
470                 FlPrintFile (FileId, "%*s}\n", Index, " ");
471                 continue;
472             }

474             /* Ignore spaces/tabs at the start of line */

476             else if ((FileByte == ' ') && StartOfLine)
477             {
478                 continue;
479             }

481             else if (StartOfLine)
482             {
483                 StartOfLine = FALSE;
484                 FlPrintFile (FileId, "%*s", Index, " ");
485             }

487 WriteByte:
488             FlWriteFile (FileId, &FileByte, 1);
489             if (FileByte == '\n')
490             {
491                 /*
492                  * This line has been completed.
493                  * Check if an error occurred on this source line during the com
494                  * If so, we print the error message after the source line.
495                  */
496                 LsCheckException (Gbl_SourceLine, FileId);
497                 return (1);
498             }
499         }
500         else
501         {
502             FlWriteFile (FileId, &FileByte, 1);
503             if (FileByte == '\n')
504             {
505                 /*
506                  * This line has been completed.
507                  * Check if an error occurred on this source line during the com
508                  * If so, we print the error message after the source line.
509                  */
510                 LsCheckException (Gbl_SourceLine, FileId);
511                 return (1);
512             }
513         }

515     } while (FlReadFile (ASL_FILE_SOURCE_OUTPUT, &FileByte, 1) == AE_OK);

517     /* EOF on the input file was reached */

519     return (0);
520 }


523 /*****************************************************************************
```

```
 *
 * FUNCTION:    LsFlushListingBuffer
 *
 * PARAMETERS:  FileId          - ID of the listing file
 *
 * RETURN:      None
 *
 * DESCRIPTION: Flush out the current contents of the 16-byte hex AML code
 *              buffer. Usually called at the termination of a single line
 *              of source code or when the buffer is full.
 *
 ******************************************************************************/

void
LsFlushListingBuffer (
    UINT32                  FileId)
{
    UINT32                  i;


    if (Gbl_CurrentHexColumn == 0)
    {
        return;
    }

    /* Write the hex bytes */

    switch (FileId)
    {
    case ASL_FILE_LISTING_OUTPUT:

        for (i = 0; i < Gbl_CurrentHexColumn; i++)
        {
            FlPrintFile (FileId, "%2.2X ", Gbl_AmlBuffer[i]);
        }

        for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 3); i+
        {
            FlWriteFile (FileId, ".", 1);
        }

        /* Write the ASCII character associated with each of the bytes */

        LsDumpAscii (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
        break;


    case ASL_FILE_ASM_SOURCE_OUTPUT:

        for (i = 0; i < Gbl_CurrentHexColumn; i++)
        {
            if (i > 0)
            {
                FlPrintFile (FileId, ",");
            }
            FlPrintFile (FileId, "0%2.2Xh", Gbl_AmlBuffer[i]);
        }

        for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 5); i+
        {
            FlWriteFile (FileId, " ", 1);
        }

        FlPrintFile (FileId, "  ;%8.8X",
            Gbl_CurrentAmlOffset - HEX_LISTING_LINE_SIZE);
```



```
        /* Write the ASCII character associated with each of the bytes */

        LsDumpAscii (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
        break;


    case ASL_FILE_C_SOURCE_OUTPUT:

        for (i = 0; i < Gbl_CurrentHexColumn; i++)
        {
            FlPrintFile (FileId, "0x%2.2X,", Gbl_AmlBuffer[i]);
        }

        /* Pad hex output with spaces if line is shorter than max line size */

        for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 5); i+
        {
            FlWriteFile (FileId, " ", 1);
        }

        /* AML offset for the start of the line */

        FlPrintFile (FileId, "    /* %8.8X",
            Gbl_CurrentAmlOffset - Gbl_CurrentHexColumn);

        /* Write the ASCII character associated with each of the bytes */

        LsDumpAsciiInComment (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
        FlPrintFile (FileId, " */");
        break;

    default:

        /* No other types supported */

        return;
    }

    FlPrintFile (FileId, "\n");

    Gbl_CurrentHexColumn = 0;
    Gbl_HexBytesWereWritten = TRUE;
}


/*******************************************************************************
 *
 * FUNCTION:    LsPushNode
 *
 * PARAMETERS:  Filename        - Pointer to the include filename
 *
 * RETURN:      None
 *
 * DESCRIPTION: Push a listing node on the listing/include file stack. This
 *              stack enables tracking of include files (infinitely nested)
 *              and resumption of the listing of the parent file when the
 *              include file is finished.
 *
 ******************************************************************************/

void
LsPushNode (
    char                    *Filename)
{
    ASL_LISTING_NODE        *Lnode;
```

```
657     /* Create a new node */

659     Lnode = UtLocalCalloc (sizeof (ASL_LISTING_NODE));

661     /* Initialize */

663     Lnode->Filename = Filename;
664     Lnode->LineNumber = 0;

666     /* Link (push) */

668     Lnode->Next = Gbl_ListingNode;
669     Gbl_ListingNode = Lnode;
670 }


673 /*******************************************************************************
674  *
675  * FUNCTION:    LsPopNode
676  *
677  * PARAMETERS:  None
678  *
679  * RETURN:      List head after current head is popped off
680  *
681  * DESCRIPTION: Pop the current head of the list, free it, and return the
682  *              next node on the stack (the new current node).
683  *
684  ******************************************************************************/

686 ASL_LISTING_NODE *
687 LsPopNode (
688     void)
689 {
690     ASL_LISTING_NODE        *Lnode;


693     /* Just grab the node at the head of the list */

695     Lnode = Gbl_ListingNode;
696     if ((!Lnode) ||
697         (!Lnode->Next))
698     {
699         AslError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL, NULL,
700             "Could not pop empty listing stack");
701         return (Gbl_ListingNode);
702     }

704     Gbl_ListingNode = Lnode->Next;
705     ACPI_FREE (Lnode);

707     /* New "Current" node is the new head */

709     return (Gbl_ListingNode);
710 }
```

```
**********************************************************
   26830 Thu Dec 26 13:48:29 2013
new/usr/src/common/acpica/compiler/aslload.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dswload - Dispatcher namespace load callbacks
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __ASLLOAD_C__

#include "aslcompiler.h"
#include "amlcode.h"
#include "acdispat.h"
#include "acnamesp.h"

#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslload")

/* Local prototypes */

static ACPI_STATUS
LdLoadFieldElements (
    ACPI_PARSE_OBJECT       *Op,
```



```
    ACPI_WALK_STATE         *WalkState);

static ACPI_STATUS
LdLoadResourceElements (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState);

static ACPI_STATUS
LdNamespace1Begin (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
LdNamespace2Begin (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
LdCommonNamespaceEnd (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);


/******************************************************************************
 *
 * FUNCTION:    LdLoadNamespace
 *
 * PARAMETERS:  RootOp      - Root of the parse tree
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform a walk of the parse tree that in turn loads all of the
 *              named ASL/AML objects into the namespace. The namespace is
 *              constructed in order to resolve named references and references
 *              to named fields within resource templates/descriptors.
 *
 *****************************************************************************/

ACPI_STATUS
LdLoadNamespace (
    ACPI_PARSE_OBJECT       *RootOp)
{
    ACPI_WALK_STATE         *WalkState;


    DbgPrint (ASL_DEBUG_OUTPUT, "\nCreating namespace\n\n");

    /* Create a new walk state */

    WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
    if (!WalkState)
    {
        return (AE_NO_MEMORY);
    }

    /* Walk the entire parse tree, first pass */

    TrWalkParseTree (RootOp, ASL_WALK_VISIT_TWICE, LdNamespace1Begin,
        LdCommonNamespaceEnd, WalkState);

    /* Second pass to handle forward references */

    TrWalkParseTree (RootOp, ASL_WALK_VISIT_TWICE, LdNamespace2Begin,
```

```
127         LdCommonNamespaceEnd, WalkState);

129     /* Dump the namespace if debug is enabled */

131     AcpiNsDumpTables (ACPI_NS_ALL, ACPI_UINT32_MAX);
132     return (AE_OK);
133 }


136 /*******************************************************************************
137  *
138  * FUNCTION:    LdLoadFieldElements
139  *
140  * PARAMETERS:  Op              - Parent node (Field)
141  *              WalkState       - Current walk state
142  *
143  * RETURN:      Status
144  *
145  * DESCRIPTION: Enter the named elements of the field (children of the parent)
146  *              into the namespace.
147  *
148  ******************************************************************************/

150 static ACPI_STATUS
151 LdLoadFieldElements (
152     ACPI_PARSE_OBJECT       *Op,
153     ACPI_WALK_STATE         *WalkState)
154 {
155     ACPI_PARSE_OBJECT       *Child = NULL;
156     ACPI_NAMESPACE_NODE     *Node;
157     ACPI_STATUS             Status;


160     /* Get the first named field element */

162     switch (Op->Asl.AmlOpcode)
163     {
164     case AML_BANK_FIELD_OP:

166         Child = UtGetArg (Op, 6);
167         break;

169     case AML_INDEX_FIELD_OP:

171         Child = UtGetArg (Op, 5);
172         break;

174     case AML_FIELD_OP:

176         Child = UtGetArg (Op, 4);
177         break;

179     default:

181         /* No other opcodes should arrive here */

183         return (AE_BAD_PARAMETER);
184     }

186     /* Enter all elements into the namespace */

188     while (Child)
189     {
190         switch (Child->Asl.AmlOpcode)
191         {
192         case AML_INT_RESERVEDFIELD_OP:
```



```
193         case AML_INT_ACCESSFIELD_OP:
194         case AML_INT_CONNECTION_OP:
195             break;

197         default:

199             Status = AcpiNsLookup (WalkState->ScopeInfo,
200                         Child->Asl.Value.String,
201                         ACPI_TYPE_LOCAL_REGION_FIELD,
202                         ACPI_IMODE_LOAD_PASS1,
203                         ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE |
204                             ACPI_NS_ERROR_IF_FOUND,
205                         NULL, &Node);
206             if (ACPI_FAILURE (Status))
207             {
208                 if (Status != AE_ALREADY_EXISTS)
209                 {
210                     AslError (ASL_ERROR, ASL_MSG_CORE_EXCEPTION, Child,
211                         Child->Asl.Value.String);
212                     return (Status);
213                 }

215                 /*
216                  * The name already exists in this scope
217                  * But continue processing the elements
218                  */
219                 AslError (ASL_ERROR, ASL_MSG_NAME_EXISTS, Child,
220                     Child->Asl.Value.String);
221             }
222             else
223             {
224                 Child->Asl.Node = Node;
225                 Node->Op = Child;
226             }
227             break;
228         }

230         Child = Child->Asl.Next;
231     }

233     return (AE_OK);
234 }


237 /*******************************************************************************
238  *
239  * FUNCTION:    LdLoadResourceElements
240  *
241  * PARAMETERS:  Op              - Parent node (Resource Descriptor)
242  *              WalkState       - Current walk state
243  *
244  * RETURN:      Status
245  *
246  * DESCRIPTION: Enter the named elements of the resource descriptor (children
247  *              of the parent) into the namespace.
248  *
249  * NOTE: In the real AML namespace, these named elements never exist. But
250  *       we simply use the namespace here as a symbol table so we can look
251  *       them up as they are referenced.
252  *
253  ******************************************************************************/

255 static ACPI_STATUS
256 LdLoadResourceElements (
257     ACPI_PARSE_OBJECT       *Op,
258     ACPI_WALK_STATE         *WalkState)
```

```
{
    ACPI_PARSE_OBJECT       *InitializerOp = NULL;
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;


    /*
     * Enter the resource name into the namespace. Name must not already exist.
     * This opens a scope, so later field names are guaranteed to be new/unique.
     */
    Status = AcpiNsLookup (WalkState->ScopeInfo, Op->Asl.Namepath,
                ACPI_TYPE_LOCAL_RESOURCE, ACPI_IMODE_LOAD_PASS1,
                ACPI_NS_NO_UPSEARCH | ACPI_NS_ERROR_IF_FOUND,
                WalkState, &Node);
    if (ACPI_FAILURE (Status))
    {
        if (Status == AE_ALREADY_EXISTS)
        {
            /* Actual node causing the error was saved in ParentMethod */

            AslError (ASL_ERROR, ASL_MSG_NAME_EXISTS,
                (ACPI_PARSE_OBJECT *) Op->Asl.ParentMethod, Op->Asl.Namepath);
            return (AE_OK);
        }
        return (Status);
    }

    Node->Value = (UINT32) Op->Asl.Value.Integer;
    Node->Op = Op;
    Op->Asl.Node = Node;

    /*
     * Now enter the predefined fields, for easy lookup when referenced
     * by the source ASL
     */
    InitializerOp = ASL_GET_CHILD_NODE (Op);
    while (InitializerOp)
    {
        if (InitializerOp->Asl.ExternalName)
        {
            Status = AcpiNsLookup (WalkState->ScopeInfo,
                        InitializerOp->Asl.ExternalName,
                        ACPI_TYPE_LOCAL_RESOURCE_FIELD,
                        ACPI_IMODE_LOAD_PASS1,
                        ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE,
                        NULL, &Node);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            /*
             * Store the field offset and length in the namespace node
             * so it can be used when the field is referenced
             */
            Node->Value = InitializerOp->Asl.Value.Tag.BitOffset;
            Node->Length = InitializerOp->Asl.Value.Tag.BitLength;
            InitializerOp->Asl.Node = Node;
            Node->Op = InitializerOp;
        }

        InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
    }

    return (AE_OK);
}
```



```


/*******************************************************************************
 *
 * FUNCTION:    LdNamespace1Begin
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Descending callback used during the parse tree walk. If this
 *              is a named AML opcode, enter into the namespace
 *
 ******************************************************************************/

static ACPI_STATUS
LdNamespace1Begin (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_WALK_STATE         *WalkState = (ACPI_WALK_STATE *) Context;
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;
    ACPI_OBJECT_TYPE        ObjectType;
    ACPI_OBJECT_TYPE        ActualObjectType = ACPI_TYPE_ANY;
    char                    *Path;
    UINT32                  Flags = ACPI_NS_NO_UPSEARCH;
    ACPI_PARSE_OBJECT       *Arg;
    UINT32                  i;
    BOOLEAN                 ForceNewScope = FALSE;


    ACPI_FUNCTION_NAME (LdNamespace1Begin);
    ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op %p [%s]\n",
        Op, Op->Asl.ParseOpName));


    /*
     * We are only interested in opcodes that have an associated name
     * (or multiple names)
     */
    switch (Op->Asl.AmlOpcode)
    {
    case AML_BANK_FIELD_OP:
    case AML_INDEX_FIELD_OP:
    case AML_FIELD_OP:

        Status = LdLoadFieldElements (Op, WalkState);
        return (Status);

    default:

        /* All other opcodes go below */

        break;
    }

    /* Check if this object has already been installed in the namespace */

    if (Op->Asl.Node)
    {
        return (AE_OK);
    }

    Path = Op->Asl.Namepath;
```

```
391     if (!Path)
392     {
393         return (AE_OK);
394     }

396     /* Map the raw opcode into an internal object type */

398     switch (Op->Asl.ParseOpcode)
399     {
400     case PARSEOP_NAME:

402         Arg = Op->Asl.Child;  /* Get the NameSeg/NameString node */
403         Arg = Arg->Asl.Next;  /* First peer is the object to be associated with

405         /*
406          * If this name refers to a ResourceTemplate, we will need to open
407          * a new scope so that the resource subfield names can be entered into
408          * the namespace underneath this name
409          */
410         if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC)
411         {
412             ForceNewScope = TRUE;
413         }

415         /* Get the data type associated with the named object, not the name itse

417         /* Log2 loop to convert from Btype (binary) to Etype (encoded) */

419         ObjectType = 1;
420         for (i = 1; i < Arg->Asl.AcpiBtype; i *= 2)
421         {
422             ObjectType++;
423         }
424         break;


427     case PARSEOP_EXTERNAL:
428         /*
429          * "External" simply enters a name and type into the namespace.
430          * We must be careful to not open a new scope, however, no matter
431          * what type the external name refers to (e.g., a method)
432          *
433          * first child is name, next child is ObjectType
434          */
435         ActualObjectType = (UINT8) Op->Asl.Child->Asl.Next->Asl.Value.Integer;
436         ObjectType = ACPI_TYPE_ANY;

438         /*
439          * We will mark every new node along the path as "External". This
440          * allows some or all of the nodes to be created later in the ASL
441          * code. Handles cases like this:
442          *
443          *   External (\_SB_.PCI0.ABCD, IntObj)
444          *   Scope (_SB_)
445          *   {
446          *       Device (PCI0)
447          *       {
448          *       }
449          *   }
450          *   Method (X)
451          *   {
452          *       Store (\_SB_.PCI0.ABCD, Local0)
453          *   }
454          */
455         Flags |= ACPI_NS_EXTERNAL;
456         break;
```



```
458     case PARSEOP_DEFAULT_ARG:

460         if (Op->Asl.CompileFlags == NODE_IS_RESOURCE_DESC)
461         {
462             Status = LdLoadResourceElements (Op, WalkState);
463             return_ACPI_STATUS (Status);
464         }

466         ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
467         break;


470     case PARSEOP_SCOPE:
471         /*
472          * The name referenced by Scope(Name) must already exist at this point.
473          * In other words, forward references for Scope() are not supported.
474          * The only real reason for this is that the MS interpreter cannot
475          * handle this case. Perhaps someday this case can go away.
476          */
477         Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ACPI_TYPE_ANY,
478                     ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT,
479                     WalkState, &(Node));
480         if (ACPI_FAILURE (Status))
481         {
482             if (Status == AE_NOT_FOUND)
483             {
484                 /* The name was not found, go ahead and create it */

486                 Status = AcpiNsLookup (WalkState->ScopeInfo, Path,
487                             ACPI_TYPE_LOCAL_SCOPE,
488                             ACPI_IMODE_LOAD_PASS1, Flags,
489                             WalkState, &(Node));
490                 if (ACPI_FAILURE (Status))
491                 {
492                     return_ACPI_STATUS (Status);
493                 }

495                 /*
496                  * However, this is an error -- primarily because the MS
497                  * interpreter can't handle a forward reference from the
498                  * Scope() operator.
499                  */
500                 AslError (ASL_ERROR, ASL_MSG_NOT_FOUND, Op,
501                     Op->Asl.ExternalName);
502                 AslError (ASL_ERROR, ASL_MSG_SCOPE_FWD_REF, Op,
503                     Op->Asl.ExternalName);
504                 goto FinishNode;
505             }

507             AslCoreSubsystemError (Op, Status,
508                 "Failure from namespace lookup", FALSE);

510             return_ACPI_STATUS (Status);
511         }

513         /* We found a node with this name, now check the type */

515         switch (Node->Type)
516         {
517         case ACPI_TYPE_LOCAL_SCOPE:
518         case ACPI_TYPE_DEVICE:
519         case ACPI_TYPE_POWER:
520         case ACPI_TYPE_PROCESSOR:
521         case ACPI_TYPE_THERMAL:
```

```
523             /* These are acceptable types - they all open a new scope */
524             break;

526         case ACPI_TYPE_INTEGER:
527         case ACPI_TYPE_STRING:
528         case ACPI_TYPE_BUFFER:
529             /*
530              * These types we will allow, but we will change the type.
531              * This enables some existing code of the form:
532              *
533              *  Name (DEB, 0)
534              *  Scope (DEB) { ... }
535              *
536              * Which is used to workaround the fact that the MS interpreter
537              * does not allow Scope() forward references.
538              */
539             sprintf (MsgBuffer, "%s [%s], changing type to [Scope]",
540                 Op->Asl.ExternalName, AcpiUtGetTypeName (Node->Type));
541             AslError (ASL_REMARK, ASL_MSG_SCOPE_TYPE, Op, MsgBuffer);

543             /* Switch the type to scope, open the new scope */

545             Node->Type = ACPI_TYPE_LOCAL_SCOPE;
546             Status = AcpiDsScopeStackPush (Node, ACPI_TYPE_LOCAL_SCOPE,
547                         WalkState);
548             if (ACPI_FAILURE (Status))
549             {
550                 return_ACPI_STATUS (Status);
551             }
552             break;

554         default:

556             /* All other types are an error */

558             sprintf (MsgBuffer, "%s [%s]", Op->Asl.ExternalName,
559                 AcpiUtGetTypeName (Node->Type));
560             AslError (ASL_ERROR, ASL_MSG_SCOPE_TYPE, Op, MsgBuffer);

562             /*
563              * However, switch the type to be an actual scope so
564              * that compilation can continue without generating a whole
565              * cascade of additional errors. Open the new scope.
566              */
567             Node->Type = ACPI_TYPE_LOCAL_SCOPE;
568             Status = AcpiDsScopeStackPush (Node, ACPI_TYPE_LOCAL_SCOPE,
569                         WalkState);
570             if (ACPI_FAILURE (Status))
571             {
572                 return_ACPI_STATUS (Status);
573             }
574             break;
575         }

577         Status = AE_OK;
578         goto FinishNode;


581     default:

583         ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
584         break;
585     }


588     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Loading name: %s, (%s)\n",
```



```
589             Op->Asl.ExternalName, AcpiUtGetTypeName (ObjectType)));

591     /* The name must not already exist */

593     Flags |= ACPI_NS_ERROR_IF_FOUND;

595     /*
596      * Enter the named type into the internal namespace. We enter the name
597      * as we go downward in the parse tree. Any necessary subobjects that
598      * involve arguments to the opcode must be created as we go back up the
599      * parse tree later.
600      */
601     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
602                 ACPI_IMODE_LOAD_PASS1, Flags, WalkState, &Node);
603     if (ACPI_FAILURE (Status))
604     {
605         if (Status == AE_ALREADY_EXISTS)
606         {
607             /* The name already exists in this scope */

609             if (Node->Type == ACPI_TYPE_LOCAL_SCOPE)
610             {
611                 /* Allow multiple references to the same scope */

613                 Node->Type = (UINT8) ObjectType;
614                 Status = AE_OK;
615             }
616             else if ((Node->Flags & ANOBJ_IS_EXTERNAL) &&
617                      (Op->Asl.ParseOpcode != PARSEOP_EXTERNAL))
618             {
619                 /*
620                  * Allow one create on an object or segment that was
621                  * previously declared External
622                  */
623                 Node->Flags &= ~ANOBJ_IS_EXTERNAL;
624                 Node->Type = (UINT8) ObjectType;

626                 /* Just retyped a node, probably will need to open a scope */

628                 if (AcpiNsOpensScope (ObjectType))
629                 {
630                     Status = AcpiDsScopeStackPush (Node, ObjectType, WalkState);
631                     if (ACPI_FAILURE (Status))
632                     {
633                         return_ACPI_STATUS (Status);
634                     }
635                 }
636                 Status = AE_OK;
637             }
638             else
639             {
640                 /* Valid error, object already exists */

642                 AslError (ASL_ERROR, ASL_MSG_NAME_EXISTS, Op,
643                     Op->Asl.ExternalName);
644                 return_ACPI_STATUS (AE_OK);
645             }
646         }
647         else
648         {
649             AslCoreSubsystemError (Op, Status,
650                 "Failure from namespace lookup", FALSE);
651             return_ACPI_STATUS (Status);
652         }
653     }
```

```
655     if (ForceNewScope)
656     {
657         Status = AcpiDsScopeStackPush (Node, ObjectType, WalkState);
658         if (ACPI_FAILURE (Status))
659         {
660             return_ACPI_STATUS (Status);
661         }
662     }

664 FinishNode:
665     /*
666      * Point the parse node to the new namespace node, and point
667      * the Node back to the original Parse node
668      */
669     Op->Asl.Node = Node;
670     Node->Op = Op;

672     /* Set the actual data type if appropriate (EXTERNAL term only) */

674     if (ActualObjectType != ACPI_TYPE_ANY)
675     {
676         Node->Type = (UINT8) ActualObjectType;
677         Node->Value = ASL_EXTERNAL_METHOD;
678     }

680     if (Op->Asl.ParseOpcode == PARSEOP_METHOD)
681     {
682         /*
683          * Get the method argument count from "Extra" and save
684          * it in the namespace node
685          */
686         Node->Value = (UINT32) Op->Asl.Extra;
687     }

689     return_ACPI_STATUS (Status);
690 }


693 /*******************************************************************************
694  *
695  * FUNCTION:    LdNamespace2Begin
696  *
697  * PARAMETERS:  ASL_WALK_CALLBACK
698  *
699  * RETURN:      Status
700  *
701  * DESCRIPTION: Descending callback used during the pass 2 parse tree walk.
702  *              Second pass resolves some forward references.
703  *
704  * Notes:
705  * Currently only needs to handle the Alias operator.
706  * Could be used to allow forward references from the Scope() operator, but
707  * the MS interpreter does not allow this, so this compiler does not either.
708  *
709  ******************************************************************************/

711 static ACPI_STATUS
712 LdNamespace2Begin (
713     ACPI_PARSE_OBJECT       *Op,
714     UINT32                  Level,
715     void                    *Context)
716 {
717     ACPI_WALK_STATE         *WalkState = (ACPI_WALK_STATE *) Context;
718     ACPI_STATUS             Status;
719     ACPI_NAMESPACE_NODE     *Node;
720     ACPI_OBJECT_TYPE        ObjectType;
```



```
721     BOOLEAN                 ForceNewScope = FALSE;
722     ACPI_PARSE_OBJECT       *Arg;
723     char                    *Path;
724     ACPI_NAMESPACE_NODE     *TargetNode;


727     ACPI_FUNCTION_NAME (LdNamespace2Begin);
728     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op %p [%s]\n",
729         Op, Op->Asl.ParseOpName));


732     /* Ignore Ops with no namespace node */

734     Node = Op->Asl.Node;
735     if (!Node)
736     {
737         return (AE_OK);
738     }

740     /* Get the type to determine if we should push the scope */

742     if ((Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG) &&
743         (Op->Asl.CompileFlags == NODE_IS_RESOURCE_DESC))
744     {
745         ObjectType = ACPI_TYPE_LOCAL_RESOURCE;
746     }
747     else
748     {
749         ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
750     }

752     /* Push scope for Resource Templates */

754     if (Op->Asl.ParseOpcode == PARSEOP_NAME)
755     {
756         if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC)
757         {
758             ForceNewScope = TRUE;
759         }
760     }

762     /* Push the scope stack */

764     if (ForceNewScope || AcpiNsOpensScope (ObjectType))
765     {
766         Status = AcpiDsScopeStackPush (Node, ObjectType, WalkState);
767         if (ACPI_FAILURE (Status))
768         {
769             return_ACPI_STATUS (Status);
770         }
771     }

773     if (Op->Asl.ParseOpcode == PARSEOP_ALIAS)
774     {
775         /* Complete the alias node by getting and saving the target node */

777         /* First child is the alias target */

779         Arg = Op->Asl.Child;

781         /* Get the target pathname */

783         Path = Arg->Asl.Namepath;
784         if (!Path)
785         {
786             Status = UtInternalizeName (Arg->Asl.ExternalName, &Path);
```

```
787             if (ACPI_FAILURE (Status))
788             {
789                 return (Status);
790             }
791         }

793         /* Get the NS node associated with the target. It must exist. */

795         Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ACPI_TYPE_ANY,
796                     ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPE
797                     WalkState, &TargetNode);
798         if (ACPI_FAILURE (Status))
799         {
800             if (Status == AE_NOT_FOUND)
801             {
802                 AslError (ASL_ERROR, ASL_MSG_NOT_FOUND, Op,
803                     Op->Asl.ExternalName);

805                 /*
806                  * The name was not found, go ahead and create it.
807                  * This prevents more errors later.
808                  */
809                 Status = AcpiNsLookup (WalkState->ScopeInfo, Path,
810                             ACPI_TYPE_ANY,
811                             ACPI_IMODE_LOAD_PASS1, ACPI_NS_NO_UPSEARCH,
812                             WalkState, &(Node));
813                 return (AE_OK);
814             }

816             AslCoreSubsystemError (Op, Status,
817                 "Failure from namespace lookup", FALSE);
818             return (AE_OK);
819         }

821         /* Save the target node within the alias node */

823         Node->Object = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, TargetNode);
824     }

826     return (AE_OK);
827 }


830 /*******************************************************************************
831  *
832  * FUNCTION:    LdCommonNamespaceEnd
833  *
834  * PARAMETERS:  ASL_WALK_CALLBACK
835  *
836  * RETURN:      Status
837  *
838  * DESCRIPTION: Ascending callback used during the loading of the namespace,
839  *              We only need to worry about managing the scope stack here.
840  *
841  ******************************************************************************/

843 static ACPI_STATUS
844 LdCommonNamespaceEnd (
845     ACPI_PARSE_OBJECT       *Op,
846     UINT32                  Level,
847     void                    *Context)
848 {
849     ACPI_WALK_STATE         *WalkState = (ACPI_WALK_STATE *) Context;
850     ACPI_OBJECT_TYPE        ObjectType;
851     BOOLEAN                 ForceNewScope = FALSE;
```



```
854     ACPI_FUNCTION_NAME (LdCommonNamespaceEnd);


857     /* We are only interested in opcodes that have an associated name */

859     if (!Op->Asl.Namepath)
860     {
861         return (AE_OK);
862     }

864     /* Get the type to determine if we should pop the scope */

866     if ((Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG) &&
867         (Op->Asl.CompileFlags == NODE_IS_RESOURCE_DESC))
868     {
869         /* TBD: Merge into AcpiDsMapNamedOpcodeToDataType */

871         ObjectType = ACPI_TYPE_LOCAL_RESOURCE;
872     }
873     else
874     {
875         ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
876     }

878     /* Pop scope that was pushed for Resource Templates */

880     if (Op->Asl.ParseOpcode == PARSEOP_NAME)
881     {
882         if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC)
883         {
884             ForceNewScope = TRUE;
885         }
886     }

888     /* Pop the scope stack */

890     if (ForceNewScope || AcpiNsOpensScope (ObjectType))
891     {
892         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
893             "(%s): Popping scope for Op [%s] %p\n",
894             AcpiUtGetTypeName (ObjectType), Op->Asl.ParseOpName, Op));

896         (void) AcpiDsScopeStackPop (WalkState);
897     }

899     return (AE_OK);
900 }
```

```
**********************************************************
    7479 Thu Dec 26 13:48:29 2013
new/usr/src/common/acpica/compiler/asllookup.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: asllookup- Namespace lookup functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acparser.h"
#include "amlcode.h"
#include "acnamesp.h"
#include "acdispat.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asllookup")

/* Local prototypes */

static ACPI_STATUS
LkIsObjectUsed (
```



```
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_PARSE_OBJECT *
LkGetNameOp (
    ACPI_PARSE_OBJECT       *Op);


/******************************************************************************
 *
 * FUNCTION:    LkFindUnreferencedObjects
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Namespace walk to find objects that are not referenced in any
 *              way. Must be called after the namespace has been cross
 *              referenced.
 *
 *****************************************************************************/

void
LkFindUnreferencedObjects (
    void)
{

    /* Walk entire namespace from the supplied root */

    (void) AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
                ACPI_UINT32_MAX, FALSE, LkIsObjectUsed, NULL,
                NULL, NULL);
}


/******************************************************************************
 *
 * FUNCTION:    LkIsObjectUsed
 *
 * PARAMETERS:  ACPI_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check for an unreferenced namespace object and emit a warning.
 *              We have to be careful, because some types and names are
 *              typically or always unreferenced, we don't want to issue
 *              excessive warnings. Note: Names that are declared within a
 *              control method are temporary, so we always issue a remark
 *              if they are not referenced.
 *
 *****************************************************************************/

static ACPI_STATUS
LkIsObjectUsed (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue)
{
    ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandl
    ACPI_NAMESPACE_NODE     *Next;


    /* Referenced flag is set during the namespace xref */
```

```
127     if (Node->Flags & ANOBJ_IS_REFERENCED)
128     {
129         return (AE_OK);
130     }

132     if (!Node->Op)
133     {
134         return (AE_OK);
135     }

137     /* These types are typically never directly referenced, ignore them */

139     switch (Node->Type)
140     {
141     case ACPI_TYPE_DEVICE:
142     case ACPI_TYPE_PROCESSOR:
143     case ACPI_TYPE_POWER:
144     case ACPI_TYPE_THERMAL:
145     case ACPI_TYPE_LOCAL_RESOURCE:

147         return (AE_OK);

149     default:

151         break;
152     }

154     /* Determine if the name is within a control method */

156     Next = Node->Parent;
157     while (Next)
158     {
159         if (Next->Type == ACPI_TYPE_METHOD)
160         {
161             /*
162              * Name is within a method, therefore it is temporary.
163              * Issue a remark even if it is a reserved name (starts
164              * with an underscore).
165              */
166             sprintf (MsgBuffer, "Name is within method [%4.4s]",
167                 Next->Name.Ascii);
168             AslError (ASL_REMARK, ASL_MSG_NOT_REFERENCED,
169                 LkGetNameOp (Node->Op), MsgBuffer);
170             return (AE_OK);
171         }

173         Next = Next->Parent;
174     }

176     /* The name is not within a control method */

178     /*
179      * Ignore names that start with an underscore. These are the reserved
180      * ACPI names and are typically not referenced since they are meant
181      * to be called by the host OS.
182      */
183     if (Node->Name.Ascii[0] == '_')
184     {
185         return (AE_OK);
186     }

188     /*
189      * What remains is an unresolved user name that is not within a method.
190      * However, the object could be referenced via another table, so issue
191      * the warning at level 2.
```



```
192      */
193     AslError (ASL_WARNING2, ASL_MSG_NOT_REFERENCED,
194         LkGetNameOp (Node->Op), NULL);
195     return (AE_OK);
196 }


199 /*******************************************************************************
200  *
201  * FUNCTION:    LkGetNameOp
202  *
203  * PARAMETERS:  Op              - Current Op
204  *
205  * RETURN:      NameOp associated with the input op
206  *
207  * DESCRIPTION: Find the name declaration op associated with the operator
208  *
209  ******************************************************************************/

211 static ACPI_PARSE_OBJECT *
212 LkGetNameOp (
213     ACPI_PARSE_OBJECT       *Op)
214 {
215     const ACPI_OPCODE_INFO  *OpInfo;
216     ACPI_PARSE_OBJECT       *NameOp = Op;


219     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);


222     /* Get the NamePath from the appropriate place */

224     if (OpInfo->Flags & AML_NAMED)
225     {
226         /* For nearly all NAMED operators, the name reference is the first child

228         NameOp = Op->Asl.Child;
229         if (Op->Asl.AmlOpcode == AML_ALIAS_OP)
230         {
231             /*
232              * ALIAS is the only oddball opcode, the name declaration
233              * (alias name) is the second operand
234              */
235             NameOp = Op->Asl.Child->Asl.Next;
236         }
237     }
238     else if (OpInfo->Flags & AML_CREATE)
239     {
240         /* Name must appear as the last parameter */

242         NameOp = Op->Asl.Child;
243         while (!(NameOp->Asl.CompileFlags & NODE_IS_NAME_DECLARATION))
244         {
245             NameOp = NameOp->Asl.Next;
246         }
247     }

249     return (NameOp);
250 }
```

```
**********************************************************
   13086 Thu Dec 26 13:48:30 2013
new/usr/src/common/acpica/compiler/aslmain.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: aslmain - compiler main and utilities
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define _DECLARE_GLOBALS

  46 #include "aslcompiler.h"
  47 #include "acapps.h"
  48 #include "acdisasm.h"
  49 #include <signal.h>

  51 #define _COMPONENT          ACPI_COMPILER
  52         ACPI_MODULE_NAME    ("aslmain")

  54 /*
  55  * Main routine for the iASL compiler.
  56  *
  57  * Portability note: The compiler depends upon the host for command-line
  58  * wildcard support - it is not implemented locally. For example:
```



```
  59  *
  60  * Linux/Unix systems: Shell expands wildcards automatically.
  61  *
  62  * Windows: The setargv.obj module must be linked in to automatically
  63  * expand wildcards.
  64  */

  66 /* Local prototypes */

  68 static void ACPI_SYSTEM_XFACE
  69 AslSignalHandler (
  70     int                     Sig);

  72 static void
  73 AslInitialize (
  74     void);


  77 /******************************************************************************
  78  *
  79  * FUNCTION:    Usage
  80  *
  81  * PARAMETERS:  None
  82  *
  83  * RETURN:      None
  84  *
  85  * DESCRIPTION: Display option help message.
  86  *              Optional items in square brackets.
  87  *
  88  *****************************************************************************/

  90 void
  91 Usage (
  92     void)
  93 {
  94     printf ("%s\n\n", ASL_COMPLIANCE);
  95     ACPI_USAGE_HEADER ("iasl [Options] [Files]");

  97     printf ("\nGeneral:\n");
  98     ACPI_OPTION ("-@ <file>",       "Specify command file");
  99     ACPI_OPTION ("-I <dir>",        "Specify additional include directory");
 100     ACPI_OPTION ("-T <sig>|ALL|*",  "Create table template file for ACPI <Sig>")
 101     ACPI_OPTION ("-p <prefix>",     "Specify path/filename prefix for all output
 102     ACPI_OPTION ("-v",              "Display compiler version");
 103     ACPI_OPTION ("-vo",             "Enable optimization comments");
 104     ACPI_OPTION ("-vs",             "Disable signon");

 106     printf ("\nHelp:\n");
 107     ACPI_OPTION ("-h",              "This message");
 108     ACPI_OPTION ("-hc",             "Display operators allowed in constant expre
 109     ACPI_OPTION ("-hf",             "Display help for output filename generation
 110     ACPI_OPTION ("-hr",             "Display ACPI reserved method names");
 111     ACPI_OPTION ("-ht",             "Display currently supported ACPI table name

 113     printf ("\nPreprocessor:\n");
 114     ACPI_OPTION ("-D <symbol>",     "Define symbol for preprocessor use");
 115     ACPI_OPTION ("-li",             "Create preprocessed output file (*.i)");
 116     ACPI_OPTION ("-P",              "Preprocess only and create preprocessor out
 117     ACPI_OPTION ("-Pn",             "Disable preprocessor");

 119     printf ("\nErrors, Warnings, and Remarks:\n");
 120     ACPI_OPTION ("-va",             "Disable all errors/warnings/remarks");
 121     ACPI_OPTION ("-ve",             "Report only errors (ignore warnings and rem
 122     ACPI_OPTION ("-vi",             "Less verbose errors and warnings for use wi
 123     ACPI_OPTION ("-vr",             "Disable remarks");
 124     ACPI_OPTION ("-vw <messageid>", "Disable specific warning or remark");
```

```
125     ACPI_OPTION ("-w1 -w2 -w3",     "Set warning reporting level");
126     ACPI_OPTION ("-we",             "Report warnings as errors");

128     printf ("\nAML Code Generation (*.aml):\n");
129     ACPI_OPTION ("-oa",             "Disable all optimizations (compatibility mo
130     ACPI_OPTION ("-of",             "Disable constant folding");
131     ACPI_OPTION ("-oi",             "Disable integer optimization to Zero/One/On
132     ACPI_OPTION ("-on",             "Disable named reference string optimization
133     ACPI_OPTION ("-cr",             "Disable Resource Descriptor error checking"
134     ACPI_OPTION ("-in",             "Ignore NoOp operators");
135     ACPI_OPTION ("-r <revision>",   "Override table header Revision (1-255)");

137     printf ("\nOptional Source Code Output Files:\n");
138     ACPI_OPTION ("-sc -sa",         "Create source file in C or assembler (*.c o
139     ACPI_OPTION ("-ic -ia",         "Create include file in C or assembler (*.h
140     ACPI_OPTION ("-tc -ta -ts",     "Create hex AML table in C, assembler, or AS
141     ACPI_OPTION ("-so",             "Create offset table in C (*.offset.h)");

143     printf ("\nOptional Listing Files:\n");
144     ACPI_OPTION ("-l",              "Create mixed listing file (ASL source and A
145     ACPI_OPTION ("-ln",             "Create namespace file (*.nsp)");
146     ACPI_OPTION ("-ls",             "Create combined source file (expanded inclu

148     printf ("\nData Table Compiler:\n");
149     ACPI_OPTION ("-G",              "Compile custom table that contains generic
150     ACPI_OPTION ("-vt",             "Create verbose template files (full disasse

152     printf ("\nAML Disassembler:\n");
153     ACPI_OPTION ("-d  <f1 f2 ...>", "Disassemble or decode binary ACPI tables to
154     ACPI_OPTION ("",                "  (Optional, file type is automatically det
155     ACPI_OPTION ("-da <f1 f2 ...>", "Disassemble multiple tables from single nam
156     ACPI_OPTION ("-db",             "Do not translate Buffers to Resource Templa
157     ACPI_OPTION ("-dc <f1 f2 ...>", "Disassemble AML and immediately compile it"
158     ACPI_OPTION ("",                "  (Obtain DSDT from current system if no in
159     ACPI_OPTION ("-e  <f1 f2 ...>", "Include ACPI table(s) for external symbol r
160     ACPI_OPTION ("-fe <file>",      "Specify external symbol declaration file");
161     ACPI_OPTION ("-g",              "Get ACPI tables and write to files (*.dat)"
162     ACPI_OPTION ("-in",             "Ignore NoOp opcodes");
163     ACPI_OPTION ("-vt",             "Dump binary table data in hex format within

165     printf ("\nDebug Options:\n");
166     ACPI_OPTION ("-bf -bt",         "Create debug file (full or parse tree only)
167     ACPI_OPTION ("-f",              "Ignore errors, force creation of AML output
168     ACPI_OPTION ("-m <size>",       "Set internal line buffer size (in Kbytes)")
169     ACPI_OPTION ("-n",              "Parse only, no output generation");
170     ACPI_OPTION ("-ot",             "Display compile times and statistics");
171     ACPI_OPTION ("-x <level>",      "Set debug level for trace output");
172     ACPI_OPTION ("-z",              "Do not insert new compiler ID for DataTable
173 }


176 /******************************************************************************
177  *
178  * FUNCTION:    FilenameHelp
179  *
180  * PARAMETERS:  None
181  *
182  * RETURN:      None
183  *
184  * DESCRIPTION: Display help message for output filename generation
185  *
186  *****************************************************************************/

188 void
189 AslFilenameHelp (
190     void)
```



```
191 {

193     printf ("\nAML output filename generation:\n");
194     printf ("  Output filenames are generated by appending an extension to a com
195     printf ("  filename prefix. The filename prefix is obtained via one of the\n
196     printf ("  following methods (in priority order):\n");
197     printf ("    1) The -p option specifies the prefix\n");
198     printf ("    2) The prefix of the AMLFileName in the ASL Definition Block\n"
199     printf ("    3) The prefix of the input filename\n");
200     printf ("\n");
201 }


204 /******************************************************************************
205  *
206  * FUNCTION:    AslSignalHandler
207  *
208  * PARAMETERS:  Sig                 - Signal that invoked this handler
209  *
210  * RETURN:      None
211  *
212  * DESCRIPTION: Control-C handler. Delete any intermediate files and any
213  *              output files that may be left in an indeterminate state.
214  *
215  *****************************************************************************/

217 static void ACPI_SYSTEM_XFACE
218 AslSignalHandler (
219     int                     Sig)
220 {
221     UINT32                  i;


224     signal (Sig, SIG_IGN);
225     printf ("Aborting\n\n");

227     /* Close all open files */

229     Gbl_Files[ASL_FILE_PREPROCESSOR].Handle = NULL; /* the .i file is same as so

231     for (i = ASL_FILE_INPUT; i < ASL_MAX_FILE_TYPE; i++)
232     {
233         FlCloseFile (i);
234     }

236     /* Delete any output files */

238     for (i = ASL_FILE_AML_OUTPUT; i < ASL_MAX_FILE_TYPE; i++)
239     {
240         FlDeleteFile (i);
241     }

243     exit (0);
244 }


247 /******************************************************************************
248  *
249  * FUNCTION:    AslInitialize
250  *
251  * PARAMETERS:  None
252  *
253  * RETURN:      None
254  *
255  * DESCRIPTION: Initialize compiler globals
256  *
```

```
257  *****************************************************************************/

259 static void
260 AslInitialize (
261     void)
262 {
263     UINT32                  i;


266     for (i = 0; i < ASL_NUM_FILES; i++)
267     {
268         Gbl_Files[i].Handle = NULL;
269         Gbl_Files[i].Filename = NULL;
270     }

272     Gbl_Files[ASL_FILE_STDOUT].Handle   = stdout;
273     Gbl_Files[ASL_FILE_STDOUT].Filename = "STDOUT";

275     Gbl_Files[ASL_FILE_STDERR].Handle   = stderr;
276     Gbl_Files[ASL_FILE_STDERR].Filename = "STDERR";
277 }


280 /******************************************************************************
281  *
282  * FUNCTION:    main
283  *
284  * PARAMETERS:  Standard argc/argv
285  *
286  * RETURN:      Program termination code
287  *
288  * DESCRIPTION: C main routine for the Asl Compiler. Handle command line
289  *              options and begin the compile for each file on the command line
290  *
291  *****************************************************************************/

293 int ACPI_SYSTEM_XFACE
294 main (
295     int                     argc,
296     char                    **argv)
297 {
298     ACPI_STATUS             Status;
299     int                     Index1;
300     int                     Index2;


303     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

305     /* Initialize preprocessor and compiler before command line processing */

307     signal (SIGINT, AslSignalHandler);
308     AcpiGbl_ExternalFileList = NULL;
309     AcpiDbgLevel = 0;
310     PrInitializePreprocessor ();
311     AslInitialize ();

313     Index1 = Index2 = AslCommandLine (argc, argv);

315     /* Allocate the line buffer(s), must be after command line */

317     Gbl_LineBufferSize /= 2;
318     UtExpandLineBuffers ();

320     /* Perform global actions first/only */

322     if (Gbl_GetAllTables)
```



```
323     {
324         Status = AslDoOneFile (NULL);
325         if (ACPI_FAILURE (Status))
326         {
327             return (-1);
328         }
329         return (0);
330     }

332     if (Gbl_DisassembleAll)
333     {
334         while (argv[Index1])
335         {
336             Status = AcpiDmAddToExternalFileList (argv[Index1]);
337             if (ACPI_FAILURE (Status))
338             {
339                 return (-1);
340             }

342             Index1++;
343         }
344     }

346     /* Process each pathname/filename in the list, with possible wildcards */

348     while (argv[Index2])
349     {
350         /*
351          * If -p not specified, we will use the input filename as the
352          * output filename prefix
353          */
354         if (Gbl_UseDefaultAmlFilename)
355         {
356             Gbl_OutputFilenamePrefix = argv[Index2];
357             UtConvertBackslashes (Gbl_OutputFilenamePrefix);
358         }

360         Status = AslDoOneFile (argv[Index2]);
361         if (ACPI_FAILURE (Status))
362         {
363             return (-1);
364         }

366         Index2++;
367     }

369     if (AcpiGbl_ExternalFileList)
370     {
371         AcpiDmClearExternalFileList();
372     }

374     return (0);
375 }
```

```
**********************************************************
   48474 Thu Dec 26 13:48:30 2013
new/usr/src/common/acpica/compiler/aslmap.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslmap - parser to AML opcode mapping table
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "amlcode.h"
#include "acparser.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslmap")


/*******************************************************************************
 *
 * FUNCTION:    AslMapNamedOpcodeToDataType
 *
 * PARAMETERS:  Opcode              - The Named AML opcode to map
 *
 * RETURN:      The ACPI type associated with the named opcode
 *
```



```
 * DESCRIPTION: Convert a raw Named AML opcode to the associated data type.
 *              Named opcodes are a subset of the AML opcodes.
 *
 ******************************************************************************/

ACPI_OBJECT_TYPE
AslMapNamedOpcodeToDataType (
    UINT16                  Opcode)
{
    const ACPI_OPCODE_INFO  *OpInfo;


    /*
     * There are some differences from the opcode table types, we
     * catch them here.
     */
    OpInfo = AcpiPsGetOpcodeInfo (Opcode);

    if (Opcode == AML_INT_NAMEPATH_OP)
    {
        return (ACPI_TYPE_ANY);
    }

    if (Opcode == AML_INT_METHODCALL_OP)
    {
        return (ACPI_TYPE_ANY);
    }

    if (OpInfo->Flags & AML_NSOBJECT)
    {
        return (OpInfo->ObjectType);
    }

    return (ACPI_TYPE_ANY);
}


/******************************************************************************
 *
 * DATA STRUCTURE:  AslKeywordMapping
 *
 * DESCRIPTION:     Maps the ParseOpcode to the actual AML opcode. The parse
 *                  opcodes are generated from Bison, and this table must
 *                  track any additions to them.
 *
 * Each entry in the table contains the following items:
 *
 * AML opcode   - Opcode that is written to the AML file
 * Value        - Value of the object to be written (if applicable)
 * Flags        - 1) Whether this opcode opens an AML "package".
 *
 *****************************************************************************/
/*
 * TBD:
 * AccessAttrib
 * AccessType
 * AMlop for DMA?
 * ObjectType keywords
 * Register
 */

const ASL_MAPPING_ENTRY     AslKeywordMapping [] =
{
/*! [Begin] no source code translation (keep the table structure) */
```

```
/* ACCESSAS */                  OP_TABLE_ENTRY (AML_INT_ACCESSFIELD_OP,     0,
/* ACCESSATTRIB_BLOCK */        OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_BLOCK_CALL */   OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_BYTE */         OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_MULTIBYTE */    OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_QUICK */        OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_RAW_BYTES */    OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_RAW_PROCESS */  OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_SND_RCV */      OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_WORD */         OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSATTRIB_WORD_CALL */    OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_ANY */            OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_BUF */            OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_BYTE */           OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_DWORD */          OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_QWORD */          OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACCESSTYPE_WORD */           OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* ACQUIRE */                   OP_TABLE_ENTRY (AML_ACQUIRE_OP,             0,
/* ADD */                       OP_TABLE_ENTRY (AML_ADD_OP,                 0,
/* ADDRESSINGMODE_7BIT */       OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* ADDRESSINGMODE_10BIT */      OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* ADDRESSTYPE_ACPI */          OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* ADDRESSTYPE_MEMORY */        OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* ADDRESSTYPE_NVS */           OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* ADDRESSTYPE_RESERVED */      OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* ALIAS */                     OP_TABLE_ENTRY (AML_ALIAS_OP,               0,
/* AND */                       OP_TABLE_ENTRY (AML_BIT_AND_OP,             0,
/* ARG0 */                      OP_TABLE_ENTRY (AML_ARG0,                   0,
/* ARG1 */                      OP_TABLE_ENTRY (AML_ARG1,                   0,
/* ARG2 */                      OP_TABLE_ENTRY (AML_ARG2,                   0,
/* ARG3 */                      OP_TABLE_ENTRY (AML_ARG3,                   0,
/* ARG4 */                      OP_TABLE_ENTRY (AML_ARG4,                   0,
/* ARG5 */                      OP_TABLE_ENTRY (AML_ARG5,                   0,
/* ARG6 */                      OP_TABLE_ENTRY (AML_ARG6,                   0,
/* BANKFIELD */                 OP_TABLE_ENTRY (AML_BANK_FIELD_OP,          0,
/* BITSPERBYTE_EIGHT */         OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* BITSPERBYTE_FIVE */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* BITSPERBYTE_NINE */          OP_TABLE_ENTRY (AML_BYTE_OP,                4,
/* BITSPERBYTE_SEVEN */         OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* BITSPERBYTE_SIX */           OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* BREAK */                     OP_TABLE_ENTRY (AML_BREAK_OP,               0,
/* BREAKPOINT */                OP_TABLE_ENTRY (AML_BREAK_POINT_OP,         0,
/* BUFFER */                    OP_TABLE_ENTRY (AML_BUFFER_OP,              0,
/* BUSMASTERTYPE_MASTER */      OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* BUSMASTERTYPE_NOTMASTER */   OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* BYTECONST */                 OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          0,
/* CASE */                      OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* CLOCKPHASE_FIRST */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* CLOCKPHASE_SECOND */         OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* CLOCKPOLARITY_HIGH */        OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* CLOCKPOLARITY_LOW */         OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* CONCATENATE */               OP_TABLE_ENTRY (AML_CONCAT_OP,              0,
/* CONCATENATERESTEMPLATE */    OP_TABLE_ENTRY (AML_CONCAT_RES_OP,          0,
/* CONDREFOF */                 OP_TABLE_ENTRY (AML_COND_REF_OF_OP,         0,
/* CONNECTION */                OP_TABLE_ENTRY (AML_INT_CONNECTION_OP,      0,
/* CONTINUE */                  OP_TABLE_ENTRY (AML_CONTINUE_OP,            0,
/* COPY */                      OP_TABLE_ENTRY (AML_COPY_OP,                0,
/* CREATEBITFIELD */            OP_TABLE_ENTRY (AML_CREATE_BIT_FIELD_OP,    0,
/* CREATEBYTEFIELD */           OP_TABLE_ENTRY (AML_CREATE_BYTE_FIELD_OP,   0,
/* CREATEDWORDFIELD */          OP_TABLE_ENTRY (AML_CREATE_DWORD_FIELD_OP,  0,
/* CREATEFIELD */               OP_TABLE_ENTRY (AML_CREATE_FIELD_OP,        0,
/* CREATEQWORDFIELD */          OP_TABLE_ENTRY (AML_CREATE_QWORD_FIELD_OP,  0,
/* CREATEWORDFIELD */           OP_TABLE_ENTRY (AML_CREATE_WORD_FIELD_OP,   0,
/* DATABUFFER */                OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DATATABLEREGION */           OP_TABLE_ENTRY (AML_DATA_REGION_OP,         0,
/* DEBUG */                     OP_TABLE_ENTRY (AML_DEBUG_OP,               0,
```



```
/* DECODETYPE_POS */            OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* DECODETYPE_SUB */            OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* DECREMENT */                 OP_TABLE_ENTRY (AML_DECREMENT_OP,           0,
/* DEFAULT */                   OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DEFAULT_ARG */               OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DEFINITIONBLOCK */           OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DEREFOF */                   OP_TABLE_ENTRY (AML_DEREF_OF_OP,            0,
/* DEVICE */                    OP_TABLE_ENTRY (AML_DEVICE_OP,              0,
/* DEVICEPOLARITY_HIGH */       OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* DEVICEPOLARITY_LOW */        OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* DIVIDE */                    OP_TABLE_ENTRY (AML_DIVIDE_OP,              0,
/* DMA */                       OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DMATYPE_A */                 OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* DMATYPE_COMPATIBILITY */     OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* DMATYPE_B */                 OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* DMATYPE_F */                 OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* DWORDCONST */                OP_TABLE_ENTRY (AML_RAW_DATA_DWORD,         0,
/* DWORDIO */                   OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DWORDMEMORY */               OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* DWORDSPACE */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* EISAID */                    OP_TABLE_ENTRY (AML_DWORD_OP,               0,
/* ELSE */                      OP_TABLE_ENTRY (AML_ELSE_OP,                0,
/* ELSEIF */                    OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* ENDDEPENDENTFN */            OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* ENDIAN_BIG */                OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* ENDIAN_LITTLE */             OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* ENDTAG */                    OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* ERRORNODE */                 OP_TABLE_ENTRY (AML_NOOP_OP,                0,
/* EVENT */                     OP_TABLE_ENTRY (AML_EVENT_OP,               0,
/* EXTENDEDIO */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* EXTENDEDMEMORY */            OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* EXTENDEDSPACE */             OP_TABLE_ENTRY (AML_RAW_DATA_QWORD,         0,
/* EXTERNAL */                  OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* FATAL */                     OP_TABLE_ENTRY (AML_FATAL_OP,               0,
/* FIELD */                     OP_TABLE_ENTRY (AML_FIELD_OP,               0,
/* FINDSETLEFTBIT */            OP_TABLE_ENTRY (AML_FIND_SET_LEFT_BIT_OP,   0,
/* FINDSETRIGHTBIT */           OP_TABLE_ENTRY (AML_FIND_SET_RIGHT_BIT_OP,  0,
/* FIXEDDMA */                  OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* FIXEDIO */                   OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* FLOWCONTROL_HW */            OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* FLOWCONTROL_NONE */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* FLOWCONTROL_SW */            OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* FROMBCD */                   OP_TABLE_ENTRY (AML_FROM_BCD_OP,            0,
/* FUNCTION */                  OP_TABLE_ENTRY (AML_METHOD_OP,              0,
/* GPIOINT */                   OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* GPIOIO */                    OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* I2CSERIALBUS */              OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* IF */                        OP_TABLE_ENTRY (AML_IF_OP,                  0,
/* INCLUDE */                   OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* INCLUDE_END */               OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* INCREMENT */                 OP_TABLE_ENTRY (AML_INCREMENT_OP,           0,
/* INDEX */                     OP_TABLE_ENTRY (AML_INDEX_OP,               0,
/* INDEXFIELD */                OP_TABLE_ENTRY (AML_INDEX_FIELD_OP,         0,
/* INTEGER */                   OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* INTERRUPT */                 OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* INTLEVEL_ACTIVEBOTH */       OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* INTLEVEL_ACTIVEHIGH */       OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* INTLEVEL_ACTIVELOW */        OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* INTTYPE_EDGE */              OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* INTTYPE_LEVEL */             OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* IO */                        OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* IODECODETYPE_10 */           OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* IODECODETYPE_16 */           OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* IORESTRICT_IN */             OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* IORESTRICT_NONE */           OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* IORESTRICT_OUT */            OP_TABLE_ENTRY (AML_BYTE_OP,                2,
```

```
/* IORESTRICT_PRESERVE */       OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* IRQ */                       OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* IRQNOFLAGS */                OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* LAND */                      OP_TABLE_ENTRY (AML_LAND_OP,                0,
/* LEQUAL */                    OP_TABLE_ENTRY (AML_LEQUAL_OP,              0,
/* LGREATER */                  OP_TABLE_ENTRY (AML_LGREATER_OP,            0,
/* LGREATEREQUAL */             OP_TABLE_ENTRY (AML_LGREATEREQUAL_OP,       0,
/* LLESS */                     OP_TABLE_ENTRY (AML_LLESS_OP,               0,
/* LLESSEQUAL */                OP_TABLE_ENTRY (AML_LLESSEQUAL_OP,          0,
/* LNOT */                      OP_TABLE_ENTRY (AML_LNOT_OP,                0,
/* LNOTEQUAL */                 OP_TABLE_ENTRY (AML_LNOTEQUAL_OP,           0,
/* LOAD */                      OP_TABLE_ENTRY (AML_LOAD_OP,                0,
/* LOADTABLE */                 OP_TABLE_ENTRY (AML_LOAD_TABLE_OP,          0,
/* LOCAL0 */                    OP_TABLE_ENTRY (AML_LOCAL0,                 0,
/* LOCAL1 */                    OP_TABLE_ENTRY (AML_LOCAL1,                 0,
/* LOCAL2 */                    OP_TABLE_ENTRY (AML_LOCAL2,                 0,
/* LOCAL3 */                    OP_TABLE_ENTRY (AML_LOCAL3,                 0,
/* LOCAL4 */                    OP_TABLE_ENTRY (AML_LOCAL4,                 0,
/* LOCAL5 */                    OP_TABLE_ENTRY (AML_LOCAL5,                 0,
/* LOCAL6 */                    OP_TABLE_ENTRY (AML_LOCAL6,                 0,
/* LOCAL7 */                    OP_TABLE_ENTRY (AML_LOCAL7,                 0,
/* LOCKRULE_LOCK */             OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* LOCKRULE_NOLOCK */           OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
/* LOR */                       OP_TABLE_ENTRY (AML_LOR_OP,                 0,
/* MATCH */                     OP_TABLE_ENTRY (AML_MATCH_OP,               0,
/* MATCHTYPE_MEQ */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MATCHTYPE_MGE */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MATCHTYPE_MGT */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MATCHTYPE_MLE */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MATCHTYPE_MLT */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MATCHTYPE_MTR */             OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          MATC
/* MAXTYPE_FIXED */             OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* MAXTYPE_NOTFIXED */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* MEMORY24 */                  OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* MEMORY32 */                  OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* MEMORY32FIXED */             OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
/* MEMTYPE_CACHEABLE */         OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* MEMTYPE_NONCACHEABLE */      OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* MEMTYPE_PREFETCHABLE */      OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* MEMTYPE_WRITECOMBINING */    OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* METHOD */                    OP_TABLE_ENTRY (AML_METHOD_OP,              0,
/* METHODCALL */                OP_TABLE_ENTRY (AML_INT_METHODCALL_OP,      0,
/* MID */                       OP_TABLE_ENTRY (AML_MID_OP,                 0,
/* MINTYPE_FIXED */             OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* MINTYPE_NOTFIXED */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* MOD */                       OP_TABLE_ENTRY (AML_MOD_OP,                 0,
/* MULTIPLY */                  OP_TABLE_ENTRY (AML_MULTIPLY_OP,            0,
/* MUTEX */                     OP_TABLE_ENTRY (AML_MUTEX_OP,               0,
/* NAME */                      OP_TABLE_ENTRY (AML_NAME_OP,                0,
/* NAMESEG */                   OP_TABLE_ENTRY (AML_INT_NAMEPATH_OP,        0,
/* NAMESTRING */                OP_TABLE_ENTRY (AML_INT_NAMEPATH_OP,        0,
/* NAND */                      OP_TABLE_ENTRY (AML_BIT_NAND_OP,            0,
/* NOOP */                      OP_TABLE_ENTRY (AML_NOOP_OP,                0,
/* NOR */                       OP_TABLE_ENTRY (AML_BIT_NOR_OP,             0,
/* NOT */                       OP_TABLE_ENTRY (AML_BIT_NOT_OP,             0,
/* NOTIFY */                    OP_TABLE_ENTRY (AML_NOTIFY_OP,              0,
/* OBJECTTYPE */                OP_TABLE_ENTRY (AML_TYPE_OP,                0,
/* OBJECTTYPE_BFF */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_BUF */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_DDB */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_DEV */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_EVT */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_FLD */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_INT */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_MTH */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_MTX */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
```



```
/* OBJECTTYPE_OPR */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_PKG */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_POW */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_PRO */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_STR */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_THZ */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OBJECTTYPE_UNK */            OP_TABLE_ENTRY (AML_BYTE_OP,                ACPI
/* OFFSET */                    OP_TABLE_ENTRY (AML_INT_RESERVEDFIELD_OP,   0,
/* ONE */                       OP_TABLE_ENTRY (AML_ONE_OP,                 0,
/* ONES */                      OP_TABLE_ENTRY (AML_ONES_OP,                0,
/* OPERATIONREGION */           OP_TABLE_ENTRY (AML_REGION_OP,              0,
/* OR */                        OP_TABLE_ENTRY (AML_BIT_OR_OP,              0,
/* PACKAGE */                   OP_TABLE_ENTRY (AML_PACKAGE_OP,             0,
/* PACKAGEP_LENGTH */           OP_TABLE_ENTRY (AML_PACKAGE_LENGTH,         0,
/* PARITYTYPE_EVEN */           OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* PARITYTYPE_MARK */           OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* PARITYTYPE_NONE */           OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* PARITYTYPE_ODD */            OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* PARITYTYPE_SPACE */          OP_TABLE_ENTRY (AML_BYTE_OP,                4,
/* PIN_NOPULL */                OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* PIN_PULLDEFAULT */           OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* PIN_PULLDOWN */              OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* PIN_PULLUP */                OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* POWERRESOURCE */             OP_TABLE_ENTRY (AML_POWER_RES_OP,           0,
/* PROCESSOR */                 OP_TABLE_ENTRY (AML_PROCESSOR_OP,           0,
/* QWORDCONST */                OP_TABLE_ENTRY (AML_RAW_DATA_QWORD,         0,
/* QWORDIO */                   OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* QWORDMEMORY */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* QWORDSPACE */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* RANGE_TYPE_ENTIRE */         OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* RANGE_TYPE_ISAONLY */        OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* RANGE_TYPE_NONISAONLY */     OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* RAW_DATA */                  OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* READWRITETYPE_BOTH */        OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* READWRITETYPE_READONLY */    OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* REFOF */                     OP_TABLE_ENTRY (AML_REF_OF_OP,              0,
/* REGIONSPACE_CMOS */          OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_EC */            OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_FFIXEDHW */      OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_GPIO */          OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_GSBUS */         OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_IO */            OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_IPMI */          OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_MEM */           OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_PCC */           OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_PCI */           OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_PCIBAR */        OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGIONSPACE_SMBUS */         OP_TABLE_ENTRY (AML_RAW_DATA_BYTE,          ACPI
/* REGISTER */                  OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* RELEASE */                   OP_TABLE_ENTRY (AML_RELEASE_OP,             0,
/* RESERVED_BYTES */            OP_TABLE_ENTRY (AML_INT_RESERVEDFIELD_OP,   0,
/* RESET */                     OP_TABLE_ENTRY (AML_RESET_OP,               0,
/* RESOURCETEMPLATE */          OP_TABLE_ENTRY (AML_BUFFER_OP,              0,
/* RESOURCETYPE_CONSUMER */     OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* RESOURCETYPE_PRODUCER */     OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* RETURN */                    OP_TABLE_ENTRY (AML_RETURN_OP,              0,
/* REVISION */                  OP_TABLE_ENTRY (AML_REVISION_OP,            0,
/* SCOPE */                     OP_TABLE_ENTRY (AML_SCOPE_OP,               0,
/* SERIALIZERULE_NOTSERIAL */   OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* SERIALIZERULE_SERIAL */      OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* SHARETYPE_EXCLUSIVE */       OP_TABLE_ENTRY (AML_BYTE_OP,                0,
/* SHARETYPE_EXCLUSIVEWAKE */   OP_TABLE_ENTRY (AML_BYTE_OP,                2,
/* SHARETYPE_SHARED */          OP_TABLE_ENTRY (AML_BYTE_OP,                1,
/* SHARETYPE_SHAREDWAKE */      OP_TABLE_ENTRY (AML_BYTE_OP,                3,
/* SHIFTLEFT */                 OP_TABLE_ENTRY (AML_SHIFT_LEFT_OP,          0,
/* SHIFTRIGHT */                OP_TABLE_ENTRY (AML_SHIFT_RIGHT_OP,         0,
```

```
 391 /* SIGNAL */                    OP_TABLE_ENTRY (AML_SIGNAL_OP,              0,
 392 /* SIZEOF */                    OP_TABLE_ENTRY (AML_SIZE_OF_OP,             0,
 393 /* SLAVEMODE_CONTROLLERINIT */  OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 394 /* SLAVEMODE_DEVICEINIT */      OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 395 /* SLEEP */                     OP_TABLE_ENTRY (AML_SLEEP_OP,               0,
 396 /* SPISERIALBUS */              OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
 397 /* STALL */                     OP_TABLE_ENTRY (AML_STALL_OP,               0,
 398 /* STARTDEPENDENTFN */          OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 399 /* STARTDEPENDENTFN_NOPRI */    OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 400 /* STOPBITS_ONE */              OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 401 /* STOPBITS_ONEPLUSHALF */      OP_TABLE_ENTRY (AML_BYTE_OP,                2,
 402 /* STOPBITS_TWO */              OP_TABLE_ENTRY (AML_BYTE_OP,                3,
 403 /* STOPBITS_ZERO */             OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 404 /* STORE */                     OP_TABLE_ENTRY (AML_STORE_OP,               0,
 405 /* STRING_LITERAL */            OP_TABLE_ENTRY (AML_STRING_OP,              0,
 406 /* SUBTRACT */                  OP_TABLE_ENTRY (AML_SUBTRACT_OP,            0,
 407 /* SWITCH */                    OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
 408 /* THERMALZONE */               OP_TABLE_ENTRY (AML_THERMAL_ZONE_OP,        0,
 409 /* TIMER */                     OP_TABLE_ENTRY (AML_TIMER_OP,               0,
 410 /* TOBCD */                     OP_TABLE_ENTRY (AML_TO_BCD_OP,              0,
 411 /* TOBUFFER */                  OP_TABLE_ENTRY (AML_TO_BUFFER_OP,           0,
 412 /* TODECIMALSTRING */           OP_TABLE_ENTRY (AML_TO_DECSTRING_OP,        0,
 413 /* TOHEXSTRING */               OP_TABLE_ENTRY (AML_TO_HEXSTRING_OP,        0,
 414 /* TOINTEGER */                 OP_TABLE_ENTRY (AML_TO_INTEGER_OP,          0,
 415 /* TOSTRING */                  OP_TABLE_ENTRY (AML_TO_STRING_OP,           0,
 416 /* TOUUID */                    OP_TABLE_ENTRY (AML_DWORD_OP,               0,
 417 /* TRANSLATIONTYPE_DENSE */     OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 418 /* TRANSLATIONTYPE_SPARSE */    OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 419 /* TYPE_STATIC */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 420 /* TYPE_TRANSLATION */          OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 421 /* UART_SERIALBUS */            OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP,         0,
 422 /* UNICODE */                   OP_TABLE_ENTRY (AML_BUFFER_OP,              0,
 423 /* UNLOAD */                    OP_TABLE_ENTRY (AML_UNLOAD_OP,              0,
 424 /* UPDATERULE_ONES */           OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
 425 /* UPDATERULE_PRESERVE */       OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
 426 /* UPDATERULE_ZEROS */          OP_TABLE_ENTRY (AML_BYTE_OP,                AML_
 427 /* VAR_PACKAGE */               OP_TABLE_ENTRY (AML_VAR_PACKAGE_OP,         0,
 428 /* VENDORLONG */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 429 /* VENDORSHORT */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 430 /* WAIT */                      OP_TABLE_ENTRY (AML_WAIT_OP,                0,
 431 /* WHILE */                     OP_TABLE_ENTRY (AML_WHILE_OP,               0,
 432 /* WIREMODE_FOUR */             OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 433 /* WIREMODE_THREE */            OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 434 /* WORDBUSNUMBER */             OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 435 /* WORDCONST */                 OP_TABLE_ENTRY (AML_RAW_DATA_WORD,          0,
 436 /* WORDIO */                    OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 437 /* WORDSPACE */                 OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 438 /* XFERSIZE_8 */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 439 /* XFERSIZE_16 */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 440 /* XFERSIZE_32 */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 441 /* XFERSIZE_64 */               OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 442 /* XFERSIZE_128 */              OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 443 /* XFERSIZE_256 */              OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 444 /* XFERTYPE_8 */                OP_TABLE_ENTRY (AML_BYTE_OP,                0,
 445 /* XFERTYPE_8_16 */             OP_TABLE_ENTRY (AML_BYTE_OP,                1,
 446 /* XFERTYPE_16 */               OP_TABLE_ENTRY (AML_BYTE_OP,                2,
 447 /* XOR */                       OP_TABLE_ENTRY (AML_BIT_XOR_OP,             0,
 448 /* ZERO */                      OP_TABLE_ENTRY (AML_ZERO_OP,                0,

 450 /*! [End] no source code translation !*/

 452 };
```

```
**********************************************************
   21841 Thu Dec 26 13:48:30 2013
new/usr/src/common/acpica/compiler/aslmessages.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslmessages.h - Compiler error/warning messages
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ASLMESSAGES_H
#define __ASLMESSAGES_H


typedef enum
{
    ASL_OPTIMIZATION = 0,
    ASL_REMARK,
    ASL_WARNING,
    ASL_WARNING2,
    ASL_WARNING3,
    ASL_ERROR,
    ASL_NUM_REPORT_LEVELS

} ASL_MESSAGE_TYPES;
```



```
#ifdef ASL_EXCEPTIONS

/* Strings for message reporting levels, must match values above */

const char              *AslErrorLevel [ASL_NUM_REPORT_LEVELS] = {
    "Optimize",
    "Remark  ",
    "Warning ",
    "Warning ",
    "Warning ",
    "Error   "
};

/* All lowercase versions for IDEs */

const char              *AslErrorLevelIde [ASL_NUM_REPORT_LEVELS] = {
    "optimize",
    "remark  ",
    "warning ",
    "warning ",
    "warning ",
    "error   "
};

#define ASL_ERROR_LEVEL_LENGTH          8       /* Length of strings above */
#endif

/*
 * Values for all compiler messages.
 *
 * NOTE: With the introduction of the -vw option to disable specific messages,
 * new messages should only be added to the end of this list, so that values
 * for existing messages are not disturbed.
 */
typedef enum
{
    ASL_MSG_RESERVED = 0,

    ASL_MSG_ALIGNMENT,
    ASL_MSG_ALPHANUMERIC_STRING,
    ASL_MSG_AML_NOT_IMPLEMENTED,
    ASL_MSG_ARG_COUNT_HI,
    ASL_MSG_ARG_COUNT_LO,
    ASL_MSG_ARG_INIT,
    ASL_MSG_BACKWARDS_OFFSET,
    ASL_MSG_BUFFER_LENGTH,
    ASL_MSG_CLOSE,
    ASL_MSG_COMPILER_INTERNAL,
    ASL_MSG_COMPILER_RESERVED,
    ASL_MSG_CONNECTION_MISSING,
    ASL_MSG_CONNECTION_INVALID,
    ASL_MSG_CONSTANT_EVALUATION,
    ASL_MSG_CONSTANT_FOLDED,
    ASL_MSG_CORE_EXCEPTION,
    ASL_MSG_DEBUG_FILE_OPEN,
    ASL_MSG_DEBUG_FILENAME,
    ASL_MSG_DEPENDENT_NESTING,
    ASL_MSG_DMA_CHANNEL,
    ASL_MSG_DMA_LIST,
    ASL_MSG_DUPLICATE_CASE,
    ASL_MSG_DUPLICATE_ITEM,
    ASL_MSG_EARLY_EOF,
    ASL_MSG_ENCODING_LENGTH,
    ASL_MSG_EX_INTERRUPT_LIST,
    ASL_MSG_EX_INTERRUPT_LIST_MIN,
```

```
126     ASL_MSG_EX_INTERRUPT_NUMBER,
127     ASL_MSG_FIELD_ACCESS_WIDTH,
128     ASL_MSG_FIELD_UNIT_ACCESS_WIDTH,
129     ASL_MSG_FIELD_UNIT_OFFSET,
130     ASL_MSG_GPE_NAME_CONFLICT,
131     ASL_MSG_HID_LENGTH,
132     ASL_MSG_HID_PREFIX,
133     ASL_MSG_HID_SUFFIX,
134     ASL_MSG_INCLUDE_FILE_OPEN,
135     ASL_MSG_INPUT_FILE_OPEN,
136     ASL_MSG_INTEGER_LENGTH,
137     ASL_MSG_INTEGER_OPTIMIZATION,
138     ASL_MSG_INTERRUPT_LIST,
139     ASL_MSG_INTERRUPT_NUMBER,
140     ASL_MSG_INVALID_ACCESS_SIZE,
141     ASL_MSG_INVALID_ADDR_FLAGS,
142     ASL_MSG_INVALID_CONSTANT_OP,
143     ASL_MSG_INVALID_EISAID,
144     ASL_MSG_INVALID_ESCAPE,
145     ASL_MSG_INVALID_GRAN_FIXED,
146     ASL_MSG_INVALID_GRANULARITY,
147     ASL_MSG_INVALID_LENGTH,
148     ASL_MSG_INVALID_LENGTH_FIXED,
149     ASL_MSG_INVALID_MIN_MAX,
150     ASL_MSG_INVALID_OPERAND,
151     ASL_MSG_INVALID_PERFORMANCE,
152     ASL_MSG_INVALID_PRIORITY,
153     ASL_MSG_INVALID_STRING,
154     ASL_MSG_INVALID_TARGET,
155     ASL_MSG_INVALID_TIME,
156     ASL_MSG_INVALID_TYPE,
157     ASL_MSG_INVALID_UUID,
158     ASL_MSG_ISA_ADDRESS,
159     ASL_MSG_LEADING_ASTERISK,
160     ASL_MSG_LIST_LENGTH_LONG,
161     ASL_MSG_LIST_LENGTH_SHORT,
162     ASL_MSG_LISTING_FILE_OPEN,
163     ASL_MSG_LISTING_FILENAME,
164     ASL_MSG_LOCAL_INIT,
165     ASL_MSG_LOCAL_OUTSIDE_METHOD,
166     ASL_MSG_LONG_LINE,
167     ASL_MSG_MEMORY_ALLOCATION,
168     ASL_MSG_MISSING_ENDDEPENDENT,
169     ASL_MSG_MISSING_STARTDEPENDENT,
170     ASL_MSG_MULTIPLE_DEFAULT,
171     ASL_MSG_MULTIPLE_TYPES,
172     ASL_MSG_NAME_EXISTS,
173     ASL_MSG_NAME_OPTIMIZATION,
174     ASL_MSG_NAMED_OBJECT_IN_WHILE,
175     ASL_MSG_NESTED_COMMENT,
176     ASL_MSG_NO_CASES,
177     ASL_MSG_NO_REGION,
178     ASL_MSG_NO_RETVAL,
179     ASL_MSG_NO_WHILE,
180     ASL_MSG_NON_ASCII,
181     ASL_MSG_NON_ZERO,
182     ASL_MSG_NOT_EXIST,
183     ASL_MSG_NOT_FOUND,
184     ASL_MSG_NOT_METHOD,
185     ASL_MSG_NOT_PARAMETER,
186     ASL_MSG_NOT_REACHABLE,
187     ASL_MSG_NOT_REFERENCED,
188     ASL_MSG_NULL_DESCRIPTOR,
189     ASL_MSG_NULL_STRING,
190     ASL_MSG_OPEN,
191     ASL_MSG_OUTPUT_FILE_OPEN,
```



```
192     ASL_MSG_OUTPUT_FILENAME,
193     ASL_MSG_PACKAGE_LENGTH,
194     ASL_MSG_PREPROCESSOR_FILENAME,
195     ASL_MSG_READ,
196     ASL_MSG_RECURSION,
197     ASL_MSG_REGION_BUFFER_ACCESS,
198     ASL_MSG_REGION_BYTE_ACCESS,
199     ASL_MSG_RESERVED_ARG_COUNT_HI,
200     ASL_MSG_RESERVED_ARG_COUNT_LO,
201     ASL_MSG_RESERVED_METHOD,
202     ASL_MSG_RESERVED_NO_RETURN_VAL,
203     ASL_MSG_RESERVED_OPERAND_TYPE,
204     ASL_MSG_RESERVED_PACKAGE_LENGTH,
205     ASL_MSG_RESERVED_RETURN_VALUE,
206     ASL_MSG_RESERVED_USE,
207     ASL_MSG_RESERVED_WORD,
208     ASL_MSG_RESOURCE_FIELD,
209     ASL_MSG_RESOURCE_INDEX,
210     ASL_MSG_RESOURCE_LIST,
211     ASL_MSG_RESOURCE_SOURCE,
212     ASL_MSG_RESULT_NOT_USED,
213     ASL_MSG_RETURN_TYPES,
214     ASL_MSG_SCOPE_FWD_REF,
215     ASL_MSG_SCOPE_TYPE,
216     ASL_MSG_SEEK,
217     ASL_MSG_SERIALIZED,
218     ASL_MSG_SERIALIZED_REQUIRED,
219     ASL_MSG_SINGLE_NAME_OPTIMIZATION,
220     ASL_MSG_SOME_NO_RETVAL,
221     ASL_MSG_STRING_LENGTH,
222     ASL_MSG_SWITCH_TYPE,
223     ASL_MSG_SYNC_LEVEL,
224     ASL_MSG_SYNTAX,
225     ASL_MSG_TABLE_SIGNATURE,
226     ASL_MSG_TAG_LARGER,
227     ASL_MSG_TAG_SMALLER,
228     ASL_MSG_TIMEOUT,
229     ASL_MSG_TOO_MANY_TEMPS,
230     ASL_MSG_TRUNCATION,
231     ASL_MSG_UNKNOWN_RESERVED_NAME,
232     ASL_MSG_UNREACHABLE_CODE,
233     ASL_MSG_UNSUPPORTED,
234     ASL_MSG_UPPER_CASE,
235     ASL_MSG_VENDOR_LIST,
236     ASL_MSG_WRITE,
237     ASL_MSG_RANGE,
238     ASL_MSG_BUFFER_ALLOCATION,

240     /* These messages are used by the Preprocessor only */

242     ASL_MSG_DIRECTIVE_SYNTAX,
243     ASL_MSG_ENDIF_MISMATCH,
244     ASL_MSG_ERROR_DIRECTIVE,
245     ASL_MSG_EXISTING_NAME,
246     ASL_MSG_INVALID_INVOCATION,
247     ASL_MSG_MACRO_SYNTAX,
248     ASL_MSG_TOO_MANY_ARGUMENTS,
249     ASL_MSG_UNKNOWN_DIRECTIVE,
250     ASL_MSG_UNKNOWN_PRAGMA,
251     ASL_MSG_WARNING_DIRECTIVE,

253     /* These messages are used by the data table compiler only */

255     ASL_MSG_BUFFER_ELEMENT,
256     ASL_MSG_DIVIDE_BY_ZERO,
257     ASL_MSG_FLAG_VALUE,
```

```
258     ASL_MSG_INTEGER_SIZE,
259     ASL_MSG_INVALID_EXPRESSION,
260     ASL_MSG_INVALID_FIELD_NAME,
261     ASL_MSG_INVALID_HEX_INTEGER,
262     ASL_MSG_OEM_TABLE,
263     ASL_MSG_RESERVED_VALUE,
264     ASL_MSG_UNKNOWN_LABEL,
265     ASL_MSG_UNKNOWN_SUBTABLE,
266     ASL_MSG_UNKNOWN_TABLE,
267     ASL_MSG_ZERO_VALUE

269 } ASL_MESSAGE_IDS;


272 #ifdef ASL_EXCEPTIONS

274 /*
275  * Actual message strings for each compiler message.
276  *
277  * NOTE: With the introduction of the -vw option to disable specific messages,
278  * new messages should only be added to the end of this list, so that values
279  * for existing messages are not disturbed.
280  */
281 char                        *AslMessages [] =
282 {
283 /*    The zeroth message is reserved */    "",
284 /*    ASL_MSG_ALIGNMENT */                  "Must be a multiple of alignment/gra
285 /*    ASL_MSG_ALPHANUMERIC_STRING */        "String must be entirely alphanumeri
286 /*    ASL_MSG_AML_NOT_IMPLEMENTED */        "Opcode is not implemented in compil
287 /*    ASL_MSG_ARG_COUNT_HI */               "Too many arguments",
288 /*    ASL_MSG_ARG_COUNT_LO */               "Too few arguments",
289 /*    ASL_MSG_ARG_INIT */                   "Method argument is not initialized"
290 /*    ASL_MSG_BACKWARDS_OFFSET */           "Invalid backwards offset",
291 /*    ASL_MSG_BUFFER_LENGTH */              "Effective AML buffer length is zero
292 /*    ASL_MSG_CLOSE */                      "Could not close file",
293 /*    ASL_MSG_COMPILER_INTERNAL */          "Internal compiler error",
294 /*    ASL_MSG_COMPILER_RESERVED */          "Use of compiler reserved name",
295 /*    ASL_MSG_CONNECTION_MISSING */         "A Connection operator is required f
296 /*    ASL_MSG_CONNECTION_INVALID */         "Invalid OpRegion SpaceId for use of
297 /*    ASL_MSG_CONSTANT_EVALUATION */        "Could not evaluate constant express
298 /*    ASL_MSG_CONSTANT_FOLDED */            "Constant expression evaluated and r
299 /*    ASL_MSG_CORE_EXCEPTION */             "From ACPI CA Subsystem",
300 /*    ASL_MSG_DEBUG_FILE_OPEN */            "Could not open debug file",
301 /*    ASL_MSG_DEBUG_FILENAME */             "Could not create debug filename",
302 /*    ASL_MSG_DEPENDENT_NESTING */          "Dependent function macros cannot be
303 /*    ASL_MSG_DMA_CHANNEL */                "Invalid DMA channel (must be 0-7)",
304 /*    ASL_MSG_DMA_LIST */                   "Too many DMA channels (8 max)",
305 /*    ASL_MSG_DUPLICATE_CASE */             "Case value already specified",
306 /*    ASL_MSG_DUPLICATE_ITEM */             "Duplicate value in list",
307 /*    ASL_MSG_EARLY_EOF */                  "Premature end-of-file reached",
308 /*    ASL_MSG_ENCODING_LENGTH */            "Package length too long to encode",
309 /*    ASL_MSG_EX_INTERRUPT_LIST */          "Too many interrupts (255 max)",
310 /*    ASL_MSG_EX_INTERRUPT_LIST_MIN */      "Too few interrupts (1 minimum requi
311 /*    ASL_MSG_EX_INTERRUPT_NUMBER */        "Invalid interrupt number (must be 3
312 /*    ASL_MSG_FIELD_ACCESS_WIDTH */         "Access width is greater than region
313 /*    ASL_MSG_FIELD_UNIT_ACCESS_WIDTH */    "Access width of Field Unit extends
314 /*    ASL_MSG_FIELD_UNIT_OFFSET */          "Field Unit extends beyond region li
315 /*    ASL_MSG_GPE_NAME_CONFLICT */          "Name conflicts with a previous GPE
316 /*    ASL_MSG_HID_LENGTH */                 "_HID string must be exactly 7 or 8
317 /*    ASL_MSG_HID_PREFIX */                 "_HID prefix must be all uppercase o
318 /*    ASL_MSG_HID_SUFFIX */                 "_HID suffix must be all hex digits"
319 /*    ASL_MSG_INCLUDE_FILE_OPEN */          "Could not open include file",
320 /*    ASL_MSG_INPUT_FILE_OPEN */            "Could not open input file",
321 /*    ASL_MSG_INTEGER_LENGTH */             "64-bit integer in 32-bit table, tru
322 /*    ASL_MSG_INTEGER_OPTIMIZATION */       "Integer optimized to single-byte AM
323 /*    ASL_MSG_INTERRUPT_LIST */             "Too many interrupts (16 max)",
```



```
324 /*    ASL_MSG_INTERRUPT_NUMBER */           "Invalid interrupt number (must be 0
325 /*    ASL_MSG_INVALID_ACCESS_SIZE */        "Invalid AccessSize (Maximum is 4 -
326 /*    ASL_MSG_INVALID_ADDR_FLAGS */         "Invalid combination of Length and M
327 /*    ASL_MSG_INVALID_CONSTANT_OP */        "Invalid operator in constant expres
328 /*    ASL_MSG_INVALID_EISAID */             "EISAID string must be of the form \
329 /*    ASL_MSG_INVALID_ESCAPE */             "Invalid or unknown escape sequence"
330 /*    ASL_MSG_INVALID_GRAN_FIXED */         "Granularity must be zero for fixed
331 /*    ASL_MSG_INVALID_GRANULARITY */        "Granularity must be zero or a power
332 /*    ASL_MSG_INVALID_LENGTH */             "Length is larger than Min/Max windo
333 /*    ASL_MSG_INVALID_LENGTH_FIXED */       "Length is not equal to fixed Min/Ma
334 /*    ASL_MSG_INVALID_MIN_MAX */            "Address Min is greater than Address
335 /*    ASL_MSG_INVALID_OPERAND */            "Invalid operand",
336 /*    ASL_MSG_INVALID_PERFORMANCE */        "Invalid performance/robustness valu
337 /*    ASL_MSG_INVALID_PRIORITY */           "Invalid priority value",
338 /*    ASL_MSG_INVALID_STRING */             "Invalid Hex/Octal Escape - Non-ASCI
339 /*    ASL_MSG_INVALID_TARGET */             "Target operand not allowed in const
340 /*    ASL_MSG_INVALID_TIME */               "Time parameter too long (255 max)",
341 /*    ASL_MSG_INVALID_TYPE */               "Invalid type",
342 /*    ASL_MSG_INVALID_UUID */               "UUID string must be of the form \"a
343 /*    ASL_MSG_ISA_ADDRESS */                "Maximum 10-bit ISA address (0x3FF)"
344 /*    ASL_MSG_LEADING_ASTERISK */           "Invalid leading asterisk",
345 /*    ASL_MSG_LIST_LENGTH_LONG */           "Initializer list longer than declar
346 /*    ASL_MSG_LIST_LENGTH_SHORT */          "Initializer list shorter than decla
347 /*    ASL_MSG_LISTING_FILE_OPEN */          "Could not open listing file",
348 /*    ASL_MSG_LISTING_FILENAME */           "Could not create listing filename",
349 /*    ASL_MSG_LOCAL_INIT */                 "Method local variable is not initia
350 /*    ASL_MSG_LOCAL_OUTSIDE_METHOD */       "Local or Arg used outside a control
351 /*    ASL_MSG_LONG_LINE */                  "Splitting long input line",
352 /*    ASL_MSG_MEMORY_ALLOCATION */          "Memory allocation failure",
353 /*    ASL_MSG_MISSING_ENDDEPENDENT */       "Missing EndDependentFn() macro in d
354 /*    ASL_MSG_MISSING_STARTDEPENDENT */     "Missing StartDependentFn() macro in
355 /*    ASL_MSG_MULTIPLE_DEFAULT */           "More than one Default statement wit
356 /*    ASL_MSG_MULTIPLE_TYPES */             "Multiple types",
357 /*    ASL_MSG_NAME_EXISTS */                "Name already exists in scope",
358 /*    ASL_MSG_NAME_OPTIMIZATION */          "NamePath optimized",
359 /*    ASL_MSG_NAMED_OBJECT_IN_WHILE */      "Creating a named object in a While
360 /*    ASL_MSG_NESTED_COMMENT */             "Nested comment found",
361 /*    ASL_MSG_NO_CASES */                   "No Case statements under Switch",
362 /*    ASL_MSG_NO_REGION */                  "_REG has no corresponding Operation
363 /*    ASL_MSG_NO_RETVAL */                  "Called method returns no value",
364 /*    ASL_MSG_NO_WHILE */                   "No enclosing While statement",
365 /*    ASL_MSG_NON_ASCII */                  "Invalid characters found in file",
366 /*    ASL_MSG_NON_ZERO */                   "Operand evaluates to zero",
367 /*    ASL_MSG_NOT_EXIST */                  "Object does not exist",
368 /*    ASL_MSG_NOT_FOUND */                  "Object not found or not accessible
369 /*    ASL_MSG_NOT_METHOD */                 "Not a control method, cannot invoke
370 /*    ASL_MSG_NOT_PARAMETER */              "Not a parameter, used as local only
371 /*    ASL_MSG_NOT_REACHABLE */              "Object is not accessible from this
372 /*    ASL_MSG_NOT_REFERENCED */             "Object is not referenced",
373 /*    ASL_MSG_NULL_DESCRIPTOR */            "Min/Max/Length/Gran are all zero, b
374 /*    ASL_MSG_NULL_STRING */                "Invalid zero-length (null) string",
375 /*    ASL_MSG_OPEN */                       "Could not open file",
376 /*    ASL_MSG_OUTPUT_FILE_OPEN */           "Could not open output AML file",
377 /*    ASL_MSG_OUTPUT_FILENAME */            "Could not create output filename",
378 /*    ASL_MSG_PACKAGE_LENGTH */             "Effective AML package length is zer
379 /*    ASL_MSG_PREPROCESSOR_FILENAME */      "Could not create preprocessor filen
380 /*    ASL_MSG_READ */                       "Could not read file",
381 /*    ASL_MSG_RECURSION */                  "Recursive method call",
382 /*    ASL_MSG_REGION_BUFFER_ACCESS */       "Host Operation Region requires Buff
383 /*    ASL_MSG_REGION_BYTE_ACCESS */         "Host Operation Region requires Byte
384 /*    ASL_MSG_RESERVED_ARG_COUNT_HI */      "Reserved method has too many argume
385 /*    ASL_MSG_RESERVED_ARG_COUNT_LO */      "Reserved method has too few argumen
386 /*    ASL_MSG_RESERVED_METHOD */            "Reserved name must be a control met
387 /*    ASL_MSG_RESERVED_NO_RETURN_VAL */     "Reserved method should not return a
388 /*    ASL_MSG_RESERVED_OPERAND_TYPE */      "Invalid object type for reserved na
389 /*    ASL_MSG_RESERVED_PACKAGE_LENGTH */    "Invalid package length for reserved
```

```
/*    ASL_MSG_RESERVED_RETURN_VALUE */      "Reserved method must return a value
/*    ASL_MSG_RESERVED_USE */               "Invalid use of reserved name",
/*    ASL_MSG_RESERVED_WORD */              "Use of reserved name",
/*    ASL_MSG_RESOURCE_FIELD */             "Resource field name cannot be used
/*    ASL_MSG_RESOURCE_INDEX */             "Missing ResourceSourceIndex (requir
/*    ASL_MSG_RESOURCE_LIST */              "Too many resource items (internal e
/*    ASL_MSG_RESOURCE_SOURCE */            "Missing ResourceSource string (requ
/*    ASL_MSG_RESULT_NOT_USED */            "Result is not used, operator has no
/*    ASL_MSG_RETURN_TYPES */               "Not all control paths return a valu
/*    ASL_MSG_SCOPE_FWD_REF */              "Forward references from Scope opera
/*    ASL_MSG_SCOPE_TYPE */                 "Existing object has invalid type fo
/*    ASL_MSG_SEEK */                       "Could not seek file",
/*    ASL_MSG_SERIALIZED */                 "Control Method marked Serialized",
/*    ASL_MSG_SERIALIZED_REQUIRED */        "Control Method should be made Seria
/*    ASL_MSG_SINGLE_NAME_OPTIMIZATION */   "NamePath optimized to NameSeg (uses
/*    ASL_MSG_SOME_NO_RETVAL */             "Called method may not always return
/*    ASL_MSG_STRING_LENGTH */              "String literal too long",
/*    ASL_MSG_SWITCH_TYPE */                "Switch expression is not a static I
/*    ASL_MSG_SYNC_LEVEL */                 "SyncLevel must be in the range 0-15
/*    ASL_MSG_SYNTAX */                     "",
/*    ASL_MSG_TABLE_SIGNATURE */            "Invalid Table Signature",
/*    ASL_MSG_TAG_LARGER */                 "ResourceTag larger than Field",
/*    ASL_MSG_TAG_SMALLER */                "ResourceTag smaller than Field",
/*    ASL_MSG_TIMEOUT */                    "Result is not used, possible operat
/*    ASL_MSG_TOO_MANY_TEMPS */             "Method requires too many temporary
/*    ASL_MSG_TRUNCATION */                 "64-bit return value will be truncat
/*    ASL_MSG_UNKNOWN_RESERVED_NAME */      "Unknown reserved name",
/*    ASL_MSG_UNREACHABLE_CODE */           "Statement is unreachable",
/*    ASL_MSG_UNSUPPORTED */                "Unsupported feature",
/*    ASL_MSG_UPPER_CASE */                 "Non-hex letters must be upper case"
/*    ASL_MSG_VENDOR_LIST */                "Too many vendor data bytes (7 max)"
/*    ASL_MSG_WRITE */                      "Could not write file",
/*    ASL_MSG_RANGE */                      "Constant out of range",
/*    ASL_MSG_BUFFER_ALLOCATION */          "Could not allocate line buffer",

/* Preprocessor */

/*    ASL_MSG_DIRECTIVE_SYNTAX */           "Invalid directive syntax",
/*    ASL_MSG_ENDIF_MISMATCH */             "Mismatched #endif",
/*    ASL_MSG_ERROR_DIRECTIVE */            "#error",
/*    ASL_MSG_EXISTING_NAME */              "Name is already defined",
/*    ASL_MSG_INVALID_INVOCATION */         "Invalid macro invocation",
/*    ASL_MSG_MACRO_SYNTAX */               "Invalid macro syntax",
/*    ASL_MSG_TOO_MANY_ARGUMENTS */         "Too many macro arguments",
/*    ASL_MSG_UNKNOWN_DIRECTIVE */          "Unknown directive",
/*    ASL_MSG_UNKNOWN_PRAGMA */             "Unknown pragma",
/*    ASL_MSG_WARNING_DIRECTIVE */          "#warning",

/* Table compiler */

/*    ASL_MSG_BUFFER_ELEMENT */             "Invalid element in buffer initializ
/*    ASL_MSG_DIVIDE_BY_ZERO */             "Expression contains divide-by-zero"
/*    ASL_MSG_FLAG_VALUE */                 "Flag value is too large",
/*    ASL_MSG_INTEGER_SIZE */               "Integer too large for target",
/*    ASL_MSG_INVALID_EXPRESSION */         "Invalid expression",
/*    ASL_MSG_INVALID_FIELD_NAME */         "Invalid Field Name",
/*    ASL_MSG_INVALID_HEX_INTEGER */        "Invalid hex integer constant",
/*    ASL_MSG_OEM_TABLE */                  "OEM table - unknown contents",
/*    ASL_MSG_RESERVED_VALUE */             "Reserved field must be zero",
/*    ASL_MSG_UNKNOWN_LABEL */              "Label is undefined",
/*    ASL_MSG_UNKNOWN_SUBTABLE */           "Unknown subtable type",
/*    ASL_MSG_UNKNOWN_TABLE */              "Unknown ACPI table signature",
/*    ASL_MSG_ZERO_VALUE */                 "Value must be non-zero"
};

#endif  /* ASL_EXCEPTIONS */
```



```
#endif  /* __ASLMESSAGES_H */
```

```
**********************************************************
   20665 Thu Dec 26 13:48:30 2013
new/usr/src/common/acpica/compiler/aslmethod.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslmethod.c - Control method analysis walk
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acparser.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslmethod")


/* Local prototypes */

void
MtCheckNamedObjectInMethod (
    ACPI_PARSE_OBJECT       *Op,
    ASL_METHOD_INFO         *MethodInfo);
```



```
/******************************************************************************
 *
 * FUNCTION:    MtMethodAnalysisWalkBegin
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Descending callback for the analysis walk. Check methods for:
 *              1) Initialized local variables
 *              2) Valid arguments
 *              3) Return types
 *
 *****************************************************************************/

ACPI_STATUS
MtMethodAnalysisWalkBegin (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ASL_ANALYSIS_WALK_INFO  *WalkInfo = (ASL_ANALYSIS_WALK_INFO *) Context;
    ASL_METHOD_INFO         *MethodInfo = WalkInfo->MethodStack;
    ACPI_PARSE_OBJECT       *Next;
    UINT32                  RegisterNumber;
    UINT32                  i;
    char                    LocalName[] = "Local0";
    char                    ArgName[] = "Arg0";
    ACPI_PARSE_OBJECT       *ArgNode;
    ACPI_PARSE_OBJECT       *NextType;
    ACPI_PARSE_OBJECT       *NextParamType;
    UINT8                   ActualArgs = 0;


    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_METHOD:

        TotalMethods++;

        /* Create and init method info */

        MethodInfo       = UtLocalCalloc (sizeof (ASL_METHOD_INFO));
        MethodInfo->Next = WalkInfo->MethodStack;
        MethodInfo->Op = Op;

        WalkInfo->MethodStack = MethodInfo;

        /* Get the name node, ignored here */

        Next = Op->Asl.Child;

        /* Get the NumArguments node */

        Next = Next->Asl.Next;
        MethodInfo->NumArguments = (UINT8)
            (((UINT8) Next->Asl.Value.Integer) & 0x07);

        /* Get the SerializeRule and SyncLevel nodes, ignored here */

        Next = Next->Asl.Next;
        MethodInfo->ShouldBeSerialized = (UINT8) Next->Asl.Value.Integer;

        Next = Next->Asl.Next;
        ArgNode = Next;
```

```
129         /* Get the ReturnType node */
131         Next = Next->Asl.Next;
133         NextType = Next->Asl.Child;
134         while (NextType)
135         {
136             /* Get and map each of the ReturnTypes */
138             MethodInfo->ValidReturnTypes |= AnMapObjTypeToBtype (NextType);
139             NextType->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
140             NextType = NextType->Asl.Next;
141         }
143         /* Get the ParameterType node */
145         Next = Next->Asl.Next;
147         NextType = Next->Asl.Child;
148         while (NextType)
149         {
150             if (NextType->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
151             {
152                 NextParamType = NextType->Asl.Child;
153                 while (NextParamType)
154                 {
155                     MethodInfo->ValidArgTypes[ActualArgs] |= AnMapObjTypeToBtype
156                     NextParamType->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
157                     NextParamType = NextParamType->Asl.Next;
158                 }
159             }
160             else
161             {
162                 MethodInfo->ValidArgTypes[ActualArgs] =
163                     AnMapObjTypeToBtype (NextType);
164                 NextType->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
165                 ActualArgs++;
166             }
168             NextType = NextType->Asl.Next;
169         }
171         if ((MethodInfo->NumArguments) &&
172             (MethodInfo->NumArguments != ActualArgs))
173         {
174             /* error: Param list did not match number of args */
175         }
177         /* Allow numarguments == 0 for Function() */
179         if ((!MethodInfo->NumArguments) && (ActualArgs))
180         {
181             MethodInfo->NumArguments = ActualArgs;
182             ArgNode->Asl.Value.Integer |= ActualArgs;
183         }
185         /*
186          * Actual arguments are initialized at method entry.
187          * All other ArgX "registers" can be used as locals, so we
188          * track their initialization.
189          */
190         for (i = 0; i < MethodInfo->NumArguments; i++)
191         {
192             MethodInfo->ArgInitialized[i] = TRUE;
193         }
```



```
194         break;
196     case PARSEOP_METHODCALL:
198         if (MethodInfo &&
199            (Op->Asl.Node == MethodInfo->Op->Asl.Node))
200         {
201             AslError (ASL_REMARK, ASL_MSG_RECURSION, Op, Op->Asl.ExternalName);
202         }
203         break;
205     case PARSEOP_LOCAL0:
206     case PARSEOP_LOCAL1:
207     case PARSEOP_LOCAL2:
208     case PARSEOP_LOCAL3:
209     case PARSEOP_LOCAL4:
210     case PARSEOP_LOCAL5:
211     case PARSEOP_LOCAL6:
212     case PARSEOP_LOCAL7:
214         if (!MethodInfo)
215         {
216             /*
217              * Local was used outside a control method, or there was an error
218              * in the method declaration.
219              */
220             AslError (ASL_REMARK, ASL_MSG_LOCAL_OUTSIDE_METHOD, Op, Op->Asl.Exte
221             return (AE_ERROR);
222         }
224         RegisterNumber = (Op->Asl.AmlOpcode & 0x000F);
226         /*
227          * If the local is being used as a target, mark the local
228          * initialized
229          */
230         if (Op->Asl.CompileFlags & NODE_IS_TARGET)
231         {
232             MethodInfo->LocalInitialized[RegisterNumber] = TRUE;
233         }
235         /*
236          * Otherwise, this is a reference, check if the local
237          * has been previously initialized.
238          *
239          * The only operator that accepts an uninitialized value is ObjectType()
240          */
241         else if ((!MethodInfo->LocalInitialized[RegisterNumber]) &&
242                  (Op->Asl.Parent->Asl.ParseOpcode != PARSEOP_OBJECTTYPE))
243         {
244             LocalName[strlen (LocalName) -1] = (char) (RegisterNumber + 0x30);
245             AslError (ASL_ERROR, ASL_MSG_LOCAL_INIT, Op, LocalName);
246         }
247         break;
249     case PARSEOP_ARG0:
250     case PARSEOP_ARG1:
251     case PARSEOP_ARG2:
252     case PARSEOP_ARG3:
253     case PARSEOP_ARG4:
254     case PARSEOP_ARG5:
255     case PARSEOP_ARG6:
257         if (!MethodInfo)
258         {
259             /*
```

```
260              * Arg was used outside a control method, or there was an error
261              * in the method declaration.
262              */
263             AslError (ASL_REMARK, ASL_MSG_LOCAL_OUTSIDE_METHOD, Op, Op->Asl.Exte
264             return (AE_ERROR);
265         }

267         RegisterNumber = (Op->Asl.AmlOpcode & 0x000F) - 8;
268         ArgName[strlen (ArgName) -1] = (char) (RegisterNumber + 0x30);

270         /*
271          * If the Arg is being used as a target, mark the local
272          * initialized
273          */
274         if (Op->Asl.CompileFlags & NODE_IS_TARGET)
275         {
276             MethodInfo->ArgInitialized[RegisterNumber] = TRUE;
277         }

279         /*
280          * Otherwise, this is a reference, check if the Arg
281          * has been previously initialized.
282          *
283          * The only operator that accepts an uninitialized value is ObjectType()
284          */
285         else if ((!MethodInfo->ArgInitialized[RegisterNumber]) &&
286                  (Op->Asl.Parent->Asl.ParseOpcode != PARSEOP_OBJECTTYPE))
287         {
288             AslError (ASL_ERROR, ASL_MSG_ARG_INIT, Op, ArgName);
289         }

291         /* Flag this arg if it is not a "real" argument to the method */

293         if (RegisterNumber >= MethodInfo->NumArguments)
294         {
295             AslError (ASL_REMARK, ASL_MSG_NOT_PARAMETER, Op, ArgName);
296         }
297         break;

299     case PARSEOP_RETURN:

301         if (!MethodInfo)
302         {
303             /*
304              * Probably was an error in the method declaration,
305              * no additional error here
306              */
307             ACPI_WARNING ((AE_INFO, "%p, No parent method", Op));
308             return (AE_ERROR);
309         }

311         /*
312          * A child indicates a possible return value. A simple Return or
313          * Return() is marked with NODE_IS_NULL_RETURN by the parser so
314          * that it is not counted as a "real" return-with-value, although
315          * the AML code that is actually emitted is Return(0). The AML
316          * definition of Return has a required parameter, so we are
317          * forced to convert a null return to Return(0).
318          */
319         if ((Op->Asl.Child) &&
320             (Op->Asl.Child->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
321             (!(Op->Asl.Child->Asl.CompileFlags & NODE_IS_NULL_RETURN)))
322         {
323             MethodInfo->NumReturnWithValue++;
324         }
325         else
```



```
326         {
327             MethodInfo->NumReturnNoValue++;
328         }
329         break;

331     case PARSEOP_BREAK:
332     case PARSEOP_CONTINUE:

334         Next = Op->Asl.Parent;
335         while (Next)
336         {
337             if (Next->Asl.ParseOpcode == PARSEOP_WHILE)
338             {
339                 break;
340             }
341             Next = Next->Asl.Parent;
342         }

344         if (!Next)
345         {
346             AslError (ASL_ERROR, ASL_MSG_NO_WHILE, Op, NULL);
347         }
348         break;

350     case PARSEOP_STALL:

352         /* We can range check if the argument is an integer */

354         if ((Op->Asl.Child->Asl.ParseOpcode == PARSEOP_INTEGER) &&
355             (Op->Asl.Child->Asl.Value.Integer > ACPI_UINT8_MAX))
356         {
357             AslError (ASL_ERROR, ASL_MSG_INVALID_TIME, Op, NULL);
358         }
359         break;

361     case PARSEOP_DEVICE:
362     case PARSEOP_EVENT:
363     case PARSEOP_MUTEX:
364     case PARSEOP_OPERATIONREGION:
365     case PARSEOP_POWERRESOURCE:
366     case PARSEOP_PROCESSOR:
367     case PARSEOP_THERMALZONE:

369         /*
370          * The first operand is a name to be created in the namespace.
371          * Check against the reserved list.
372          */
373         i = ApCheckForPredefinedName (Op, Op->Asl.NameSeg);
374         if (i < ACPI_VALID_RESERVED_NAME_MAX)
375         {
376             AslError (ASL_ERROR, ASL_MSG_RESERVED_USE, Op, Op->Asl.ExternalName)
377         }
378         break;

380     case PARSEOP_NAME:

382         /* Typecheck any predefined names statically defined with Name() */

384         ApCheckForPredefinedObject (Op, Op->Asl.NameSeg);

386         /* Special typechecking for _HID */

388         if (!ACPI_STRCMP (METHOD_NAME__HID, Op->Asl.NameSeg))
389         {
390             Next = Op->Asl.Child->Asl.Next;
391             AnCheckId (Next, ASL_TYPE_HID);
```

```
392         }
394         /* Special typechecking for _CID */
396         else if (!ACPI_STRCMP (METHOD_NAME__CID, Op->Asl.NameSeg))
397         {
398             Next = Op->Asl.Child->Asl.Next;
400             if ((Next->Asl.ParseOpcode == PARSEOP_PACKAGE) ||
401                 (Next->Asl.ParseOpcode == PARSEOP_VAR_PACKAGE))
402             {
403                 Next = Next->Asl.Child;
404                 while (Next)
405                 {
406                     AnCheckId (Next, ASL_TYPE_CID);
407                     Next = Next->Asl.Next;
408                 }
409             }
410             else
411             {
412                 AnCheckId (Next, ASL_TYPE_CID);
413             }
414         }
415         break;
417     default:
419         break;
420     }
422     /* Check for named object creation within a non-serialized method */
424     MtCheckNamedObjectInMethod (Op, MethodInfo);
425     return (AE_OK);
426 }


429 /*******************************************************************************
430  *
431  * FUNCTION:    MtCheckNamedObjectInMethod
432  *
433  * PARAMETERS:  Op                  - Current parser op
434  *              MethodInfo          - Info for method being parsed
435  *
436  * RETURN:      None
437  *
438  * DESCRIPTION: Detect if a non-serialized method is creating a named object,
439  *              which could possibly cause problems if two threads execute
440  *              the method concurrently. Emit a remark in this case.
441  *
442  ******************************************************************************/

444 void
445 MtCheckNamedObjectInMethod (
446     ACPI_PARSE_OBJECT       *Op,
447     ASL_METHOD_INFO         *MethodInfo)
448 {
449     const ACPI_OPCODE_INFO  *OpInfo;


452     /* We don't care about actual method declarations */
454     if (Op->Asl.AmlOpcode == AML_METHOD_OP)
455     {
456         return;
457     }
```



```
459     /* Determine if we are creating a named object */
461     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
462     if (OpInfo->Class == AML_CLASS_NAMED_OBJECT)
463     {
464         /*
465          * If we have a named object created within a non-serialized method,
466          * emit a remark that the method should be serialized.
467          *
468          * Reason: If a thread blocks within the method for any reason, and
469          * another thread enters the method, the method will fail because an
470          * attempt will be made to create the same object twice.
471          */
472         if (MethodInfo && !MethodInfo->ShouldBeSerialized)
473         {
474             AslError (ASL_REMARK, ASL_MSG_SERIALIZED_REQUIRED, MethodInfo->Op,
475                 "due to creation of named objects within");
477             /* Emit message only ONCE per method */
479             MethodInfo->ShouldBeSerialized = TRUE;
480         }
481     }
482 }


485 /*******************************************************************************
486  *
487  * FUNCTION:    MtMethodAnalysisWalkEnd
488  *
489  * PARAMETERS:  ASL_WALK_CALLBACK
490  *
491  * RETURN:      Status
492  *
493  * DESCRIPTION: Ascending callback for analysis walk. Complete method
494  *              return analysis.
495  *
496  ******************************************************************************/

498 ACPI_STATUS
499 MtMethodAnalysisWalkEnd (
500     ACPI_PARSE_OBJECT       *Op,
501     UINT32                  Level,
502     void                    *Context)
503 {
504     ASL_ANALYSIS_WALK_INFO  *WalkInfo = (ASL_ANALYSIS_WALK_INFO *) Context;
505     ASL_METHOD_INFO         *MethodInfo = WalkInfo->MethodStack;


508     switch (Op->Asl.ParseOpcode)
509     {
510     case PARSEOP_METHOD:
511     case PARSEOP_RETURN:
513         if (!MethodInfo)
514         {
515             printf ("No method info for method! [%s]\n", Op->Asl.Namepath);
516             AslError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL, Op,
517                 "No method info for this method");
519             CmCleanupAndExit ();
520             return (AE_AML_INTERNAL);
521         }
522         break;
```

```
524     default:
526         break;
527     }
529     switch (Op->Asl.ParseOpcode)
530     {
531     case PARSEOP_METHOD:
533         WalkInfo->MethodStack = MethodInfo->Next;
535         /*
536          * Check if there is no return statement at the end of the
537          * method AND we can actually get there -- i.e., the execution
538          * of the method can possibly terminate without a return statement.
539          */
540         if ((!AnLastStatementIsReturn (Op)) &&
541             (!(Op->Asl.CompileFlags & NODE_HAS_NO_EXIT)))
542         {
543             /*
544              * No return statement, and execution can possibly exit
545              * via this path. This is equivalent to Return ()
546              */
547             MethodInfo->NumReturnNoValue++;
548         }
550         /*
551          * Check for case where some return statements have a return value
552          * and some do not. Exit without a return statement is a return with
553          * no value
554          */
555         if (MethodInfo->NumReturnNoValue &&
556             MethodInfo->NumReturnWithValue)
557         {
558             AslError (ASL_WARNING, ASL_MSG_RETURN_TYPES, Op,
559                 Op->Asl.ExternalName);
560         }
562         /*
563          * If there are any RETURN() statements with no value, or there is a
564          * control path that allows the method to exit without a return value,
565          * we mark the method as a method that does not return a value. This
566          * knowledge can be used to check method invocations that expect a
567          * returned value.
568          */
569         if (MethodInfo->NumReturnNoValue)
570         {
571             if (MethodInfo->NumReturnWithValue)
572             {
573                 Op->Asl.CompileFlags |= NODE_METHOD_SOME_NO_RETVAL;
574             }
575             else
576             {
577                 Op->Asl.CompileFlags |= NODE_METHOD_NO_RETVAL;
578             }
579         }
581         /*
582          * Check predefined method names for correct return behavior
583          * and correct number of arguments. Also, some special checks
584          * For GPE and _REG methods.
585          */
586         if (ApCheckForPredefinedMethod (Op, MethodInfo))
587         {
588             /* Special check for two names like _L01 and _E01 in same scope */
```



```
590             ApCheckForGpeNameConflict (Op);
592             /*
593              * Special check for _REG: Must have an operation region definition
594              * within the same scope!
595              */
596             ApCheckRegMethod (Op);
597         }
599         ACPI_FREE (MethodInfo);
600         break;
602     case PARSEOP_NAME:
604          /* Special check for two names like _L01 and _E01 in same scope */
606         ApCheckForGpeNameConflict (Op);
607         break;
609     case PARSEOP_RETURN:
611         /*
612          * If the parent is a predefined method name, attempt to typecheck
613          * the return value. Only static types can be validated.
614          */
615         ApCheckPredefinedReturnValue (Op, MethodInfo);
617         /*
618          * The parent block does not "exit" and continue execution -- the
619          * method is terminated here with the Return() statement.
620          */
621         Op->Asl.Parent->Asl.CompileFlags |= NODE_HAS_NO_EXIT;
623         /* Used in the "typing" pass later */
625         Op->Asl.ParentMethod = MethodInfo->Op;
627         /*
628          * If there is a peer node after the return statement, then this
629          * node is unreachable code -- i.e., it won't be executed because of
630          * the preceding Return() statement.
631          */
632         if (Op->Asl.Next)
633         {
634             AslError (ASL_WARNING, ASL_MSG_UNREACHABLE_CODE, Op->Asl.Next, NULL)
635         }
636         break;
638     case PARSEOP_IF:
640         if ((Op->Asl.CompileFlags & NODE_HAS_NO_EXIT) &&
641             (Op->Asl.Next) &&
642             (Op->Asl.Next->Asl.ParseOpcode == PARSEOP_ELSE))
643         {
644             /*
645              * This IF has a corresponding ELSE. The IF block has no exit,
646              * (it contains an unconditional Return)
647              * mark the ELSE block to remember this fact.
648              */
649             Op->Asl.Next->Asl.CompileFlags |= NODE_IF_HAS_NO_EXIT;
650         }
651         break;
653     case PARSEOP_ELSE:
655         if ((Op->Asl.CompileFlags & NODE_HAS_NO_EXIT) &&
```

```
656             (Op->Asl.CompileFlags & NODE_IF_HAS_NO_EXIT))
657         {
658             /*
659              * This ELSE block has no exit and the corresponding IF block
660              * has no exit either. Therefore, the parent node has no exit.
661              */
662             Op->Asl.Parent->Asl.CompileFlags |= NODE_HAS_NO_EXIT;
663         }
664         break;


667     default:

669         if ((Op->Asl.CompileFlags & NODE_HAS_NO_EXIT) &&
670             (Op->Asl.Parent))
671         {
672             /* If this node has no exit, then the parent has no exit either */

674             Op->Asl.Parent->Asl.CompileFlags |= NODE_HAS_NO_EXIT;
675         }
676         break;
677     }

679     return (AE_OK);
680 }
```

```
**********************************************************
   12397 Thu Dec 26 13:48:30 2013
new/usr/src/common/acpica/compiler/aslnamesp.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslnamesp - Namespace output file generation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslnamesp")

/* Local prototypes */

static ACPI_STATUS
NsDoOneNamespaceObject (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);
```



```
static ACPI_STATUS
NsDoOnePathname (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    NsSetupNamespaceListing
 *
 * PARAMETERS:  Handle          - local file handle
 *
 * RETURN:      None
 *
 * DESCRIPTION: Set the namespace output file to the input handle
 *
 *****************************************************************************/

void
NsSetupNamespaceListing (
    void                    *Handle)
{

    Gbl_NsOutputFlag = TRUE;
    Gbl_Files[ASL_FILE_NAMESPACE_OUTPUT].Handle = Handle;
}


/******************************************************************************
 *
 * FUNCTION:    NsDisplayNamespace
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk the namespace an display information about each node
 *              in the tree. Information is written to the optional
 *              namespace output file.
 *
 *****************************************************************************/

ACPI_STATUS
NsDisplayNamespace (
    void)
{
    ACPI_STATUS             Status;


    if (!Gbl_NsOutputFlag)
    {
        return (AE_OK);
    }

    Gbl_NumNamespaceObjects = 0;

    /* File header */

    FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "Contents of ACPI Namespace\n\n");
    FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "Count  Depth    Name - Type\n\n");

    /* Walk entire namespace from the root */

    Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
```

```
128                 ACPI_UINT32_MAX, FALSE, NsDoOneNamespaceObject, NULL,
129                 NULL, NULL);

131     /* Print the full pathname for each namespace node */

133     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "\nNamespace pathnames\n\n");

135     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
136                 ACPI_UINT32_MAX, FALSE, NsDoOnePathname, NULL,
137                 NULL, NULL);

139     return (Status);
140 }


143 /*******************************************************************************
144  *
145  * FUNCTION:    NsDoOneNamespaceObject
146  *
147  * PARAMETERS:  ACPI_WALK_CALLBACK
148  *
149  * RETURN:      Status
150  *
151  * DESCRIPTION: Dump a namespace object to the namespace output file.
152  *              Called during the walk of the namespace to dump all objects.
153  *
154  ******************************************************************************/

156 static ACPI_STATUS
157 NsDoOneNamespaceObject (
158     ACPI_HANDLE             ObjHandle,
159     UINT32                  Level,
160     void                    *Context,
161     void                    **ReturnValue)
162 {
163     ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
164     ACPI_OPERAND_OBJECT     *ObjDesc;
165     ACPI_PARSE_OBJECT       *Op;


168     Gbl_NumNamespaceObjects++;

170     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "%5u  [%u]  %*s %4.4s - %s",
171         Gbl_NumNamespaceObjects, Level, (Level * 3), " ",
172         &Node->Name,
173         AcpiUtGetTypeName (Node->Type));

175     Op = Node->Op;
176     ObjDesc = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Node->Object);

178     if (!Op)
179     {
180         FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "\n");
181         return (AE_OK);
182     }


185     if ((ObjDesc) &&
186         (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) == ACPI_DESC_TYPE_OPERAND))
187     {
188         switch (Node->Type)
189         {
190         case ACPI_TYPE_INTEGER:

192             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
193                 "       [Initial Value   0x%8.8X%8.8X]",
```



```
194                 ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
195             break;

197         case ACPI_TYPE_STRING:

199             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
200                 "       [Initial Value   \"%s\"]",
201                 ObjDesc->String.Pointer);
202             break;

204         default:

206             /* Nothing to do for other types */

208             break;
209         }

211     }
212     else
213     {
214         switch (Node->Type)
215         {
216         case ACPI_TYPE_INTEGER:

218             if (Op->Asl.ParseOpcode == PARSEOP_NAME)
219             {
220                 Op = Op->Asl.Child;
221             }
222             if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)  ||
223                 (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING))
224             {
225                 Op = Op->Asl.Next;
226             }
227             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
228                 "       [Initial Value   0x%8.8X%8.8X]",
229                 ACPI_FORMAT_UINT64 (Op->Asl.Value.Integer));
230             break;

232         case ACPI_TYPE_STRING:

234             if (Op->Asl.ParseOpcode == PARSEOP_NAME)
235             {
236                 Op = Op->Asl.Child;
237             }
238             if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)  ||
239                 (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING))
240             {
241                 Op = Op->Asl.Next;
242             }
243             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
244                 "       [Initial Value   \"%s\"]",
245                 Op->Asl.Value.String);
246             break;

248         case ACPI_TYPE_LOCAL_REGION_FIELD:

250             if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)  ||
251                 (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING))
252             {
253                 Op = Op->Asl.Child;
254             }
255             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
256                 "   [Offset 0x%04X   Length 0x%04X bits]",
257                 Op->Asl.Parent->Asl.ExtraValue, (UINT32) Op->Asl.Value.Integer);
258             break;
```

```
260         case ACPI_TYPE_BUFFER_FIELD:

262             switch (Op->Asl.ParseOpcode)
263             {
264             case PARSEOP_CREATEBYTEFIELD:

266                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [BYTE  ( 8 bit)]");
267                 break;

269             case PARSEOP_CREATEDWORDFIELD:

271                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [DWORD (32 bit)]");
272                 break;

274             case PARSEOP_CREATEQWORDFIELD:

276                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [QWORD (64 bit)]");
277                 break;

279             case PARSEOP_CREATEWORDFIELD:

281                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [WORD  (16 bit)]");
282                 break;

284             case PARSEOP_CREATEBITFIELD:

286                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [BIT   ( 1 bit)]");
287                 break;

289             case PARSEOP_CREATEFIELD:

291                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "   [Arbitrary Bit Field
292                 break;

294             default:

296                 break;

298             }
299             break;

301         case ACPI_TYPE_PACKAGE:

303             if (Op->Asl.ParseOpcode == PARSEOP_NAME)
304             {
305                 Op = Op->Asl.Child;
306             }
307             if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)  ||
308                 (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING))
309             {
310                 Op = Op->Asl.Next;
311             }
312             Op = Op->Asl.Child;

314             if ((Op->Asl.ParseOpcode == PARSEOP_BYTECONST) ||
315                 (Op->Asl.ParseOpcode == PARSEOP_RAW_DATA))
316             {
317                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
318                     "       [Initial Length  0x%.2X elements]",
319                     Op->Asl.Value.Integer);
320             }
321             break;

323         case ACPI_TYPE_BUFFER:

325             if (Op->Asl.ParseOpcode == PARSEOP_NAME)
```



```
326             {
327                 Op = Op->Asl.Child;
328             }
329             if ((Op->Asl.ParseOpcode == PARSEOP_NAMESEG)  ||
330                 (Op->Asl.ParseOpcode == PARSEOP_NAMESTRING))
331             {
332                 Op = Op->Asl.Next;
333             }
334             Op = Op->Asl.Child;

336             if (Op && (Op->Asl.ParseOpcode == PARSEOP_INTEGER))
337             {
338                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
339                     "       [Initial Length  0x%.2X bytes]",
340                     Op->Asl.Value.Integer);
341             }
342             break;

344         case ACPI_TYPE_METHOD:

346             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
347                 "       [Code Length     0x%.4X bytes]",
348                 Op->Asl.AmlSubtreeLength);
349             break;

351         case ACPI_TYPE_LOCAL_RESOURCE:

353             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
354                 "  [Desc Offset     0x%.4X Bytes]", Node->Value);
355             break;

357         case ACPI_TYPE_LOCAL_RESOURCE_FIELD:

359             FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
360                 "   [Field Offset    0x%.4X Bits 0x%.4X Bytes] ",
361                 Node->Value, Node->Value / 8);

363             if (Node->Flags & ANOBJ_IS_REFERENCED)
364             {
365                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
366                     "Referenced");
367             }
368             else
369             {
370                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
371                     "Name not referenced");
372             }
373             break;

375         default:

377             /* Nothing to do for other types */

379             break;
380         }
381     }

383     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "\n");
384     return (AE_OK);
385 }


388 /*******************************************************************************
389  *
390  * FUNCTION:    NsDoOnePathname
391  *
```

```
 * PARAMETERS:  ACPI_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Print the full pathname for a namespace node.
 *
 *****************************************************************************/

static ACPI_STATUS
NsDoOnePathname (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue)
{
    ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
    ACPI_STATUS             Status;
    ACPI_BUFFER             TargetPath;


    TargetPath.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
    Status = AcpiNsHandleToPathname (Node, &TargetPath);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "%s\n", TargetPath.Pointer);
    ACPI_FREE (TargetPath.Pointer);

    return (AE_OK);
}
```

```
**********************************************************
   15308 Thu Dec 26 13:48:31 2013
new/usr/src/common/acpica/compiler/asloffset.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asloffset - Generate a C "offset table" for BIOS use.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asloffset")


/* Local prototypes */

static void
LsEmitOffsetTableEntry (
    UINT32                  FileId,
    ACPI_NAMESPACE_NODE     *Node,
    UINT32                  NamepathOffset,
    UINT32                  Offset,
    char                    *OpName,
    UINT64                  Value,
    UINT8                   AmlOpcode,
    UINT16                  ParentOpcode);


/******************************************************************************
 *
 * FUNCTION:    LsAmlOffsetWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Process one node during a offset table file generation.
 *
 * Three types of objects are currently emitted to the offset table:
 *   1) Tagged (named) resource descriptors
 *   2) Named integer objects with constant integer values
 *   3) Named package objects
 *   4) Operation Regions that have constant Offset (address) parameters
 *   5) Control methods
 *
 * The offset table allows the BIOS to dynamically update the values of these
 * objects at boot time.
 *
 *****************************************************************************/

ACPI_STATUS
LsAmlOffsetWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    UINT32                  FileId = (UINT32) ACPI_TO_INTEGER (Context);
    ACPI_NAMESPACE_NODE     *Node;
    UINT32                  Length;
    UINT32                  NamepathOffset;
    UINT32                  DataOffset;
    ACPI_PARSE_OBJECT       *NextOp;


    /* Ignore actual data blocks for resource descriptors */

    if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DATA)
    {
        return (AE_OK); /* Do NOT update the global AML offset */
    }

    /* We are only interested in named objects (have a namespace node) */

    Node = Op->Asl.Node;
    if (!Node)
    {
        Gbl_CurrentAmlOffset += Op->Asl.FinalAmlLength;
        return (AE_OK);
    }

    /* Named resource descriptor (has a descriptor tag) */

    if ((Node->Type == ACPI_TYPE_LOCAL_RESOURCE) &&
        (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC))
    {
        LsEmitOffsetTableEntry (FileId, Node, 0, Gbl_CurrentAmlOffset,
            Op->Asl.ParseOpName, 0, Op->Asl.Extra, AML_BUFFER_OP);
```

```
128         Gbl_CurrentAmlOffset += Op->Asl.FinalAmlLength;
129         return (AE_OK);
130     }

132     switch (Op->Asl.AmlOpcode)
133     {
134     case AML_NAME_OP:

136         /* Named object -- Name (NameString, DataRefObject) */

138         if (!Op->Asl.Child)
139         {
140             FlPrintFile (FileId, "%s NO CHILD!\n", MsgBuffer);
141             return (AE_OK);
142         }

144         Length = Op->Asl.FinalAmlLength;
145         NamepathOffset = Gbl_CurrentAmlOffset + Length;

147         /* Get to the NameSeg/NamePath Op (and length of the name) */

149         Op = Op->Asl.Child;

151         /* Get offset of last nameseg and the actual data */

153         NamepathOffset = Gbl_CurrentAmlOffset + Length +
154             (Op->Asl.FinalAmlLength - ACPI_NAME_SIZE);

156         DataOffset = Gbl_CurrentAmlOffset + Length +
157             Op->Asl.FinalAmlLength;

159         /* Get actual value associated with the name */

161         Op = Op->Asl.Next;
162         switch (Op->Asl.AmlOpcode)
163         {
164         case AML_BYTE_OP:
165         case AML_WORD_OP:
166         case AML_DWORD_OP:
167         case AML_QWORD_OP:

169             /* The +1 is to handle the integer size prefix (opcode) */

171             LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, (DataOffset +
172                 Op->Asl.ParseOpName, Op->Asl.Value.Integer,
173                 (UINT8) Op->Asl.AmlOpcode, AML_NAME_OP);
174             break;

176         case AML_ONE_OP:
177         case AML_ONES_OP:
178         case AML_ZERO_OP:

180             /* For these, offset will point to the opcode */

182             LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, DataOffset,
183                 Op->Asl.ParseOpName, Op->Asl.Value.Integer,
184                 (UINT8) Op->Asl.AmlOpcode, AML_NAME_OP);
185             break;

187         case AML_PACKAGE_OP:
188         case AML_VAR_PACKAGE_OP:

190             /* Get the package element count */

192             NextOp = Op->Asl.Child;
```



```
194             LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, DataOffset,
195                 Op->Asl.ParseOpName, NextOp->Asl.Value.Integer,
196                 (UINT8) Op->Asl.AmlOpcode, AML_NAME_OP);
197             break;

199          default:
200              break;
201         }

203         Gbl_CurrentAmlOffset += Length;
204         return (AE_OK);

206     case AML_REGION_OP:

208         /* OperationRegion (NameString, RegionSpace, RegionOffset, RegionLength)

210         Length = Op->Asl.FinalAmlLength;

212         /* Get the name/namepath node */

214         NextOp = Op->Asl.Child;

216         /* Get offset of last nameseg and the actual data */

218         NamepathOffset = Gbl_CurrentAmlOffset + Length +
219             (NextOp->Asl.FinalAmlLength - ACPI_NAME_SIZE);

221         DataOffset = Gbl_CurrentAmlOffset + Length +
222             (NextOp->Asl.FinalAmlLength + 1);

224         /* Get the SpaceId node, then the Offset (address) node */

226         NextOp = NextOp->Asl.Next;
227         NextOp = NextOp->Asl.Next;

229         switch (NextOp->Asl.AmlOpcode)
230         {
231         /*
232          * We are only interested in integer constants that can be changed
233          * at boot time. Note, the One/Ones/Zero opcodes are considered
234          * non-changeable, so we ignore them here.
235          */
236         case AML_BYTE_OP:
237         case AML_WORD_OP:
238         case AML_DWORD_OP:
239         case AML_QWORD_OP:

241             LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, (DataOffset +
242                 Op->Asl.ParseOpName, NextOp->Asl.Value.Integer,
243                 (UINT8) NextOp->Asl.AmlOpcode, AML_REGION_OP);

245             Gbl_CurrentAmlOffset += Length;
246             return (AE_OK);

248         default:
249             break;
250         }
251         break;

253     case AML_METHOD_OP:

255         /* Method (Namepath, ...) */

257         Length = Op->Asl.FinalAmlLength;

259         /* Get the NameSeg/NamePath Op */
```

```
261         NextOp = Op->Asl.Child;

263         /* Get offset of last nameseg and the actual data (flags byte) */

265         NamepathOffset = Gbl_CurrentAmlOffset + Length +
266             (NextOp->Asl.FinalAmlLength - ACPI_NAME_SIZE);

268         DataOffset = Gbl_CurrentAmlOffset + Length +
269             NextOp->Asl.FinalAmlLength;

271         /* Get the flags byte Op */

273         NextOp = NextOp->Asl.Next;

275         LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, DataOffset,
276             Op->Asl.ParseOpName, NextOp->Asl.Value.Integer,
277             (UINT8) Op->Asl.AmlOpcode, AML_METHOD_OP);
278         break;

280     case AML_PROCESSOR_OP:

282         /* Processor (Namepath, ProcessorId, Address, Length) */

284         Length = Op->Asl.FinalAmlLength;
285         NextOp = Op->Asl.Child;     /* Get Namepath */

287         /* Get offset of last nameseg and the actual data (PBlock address) */

289         NamepathOffset = Gbl_CurrentAmlOffset + Length +
290             (NextOp->Asl.FinalAmlLength - ACPI_NAME_SIZE);

292         DataOffset = Gbl_CurrentAmlOffset + Length +
293             (NextOp->Asl.FinalAmlLength + 1);

295         NextOp = NextOp->Asl.Next;  /* Get ProcessorID (BYTE) */
296         NextOp = NextOp->Asl.Next;  /* Get Address (DWORD) */

298         LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, DataOffset,
299             Op->Asl.ParseOpName, NextOp->Asl.Value.Integer,
300             (UINT8) AML_DWORD_OP, AML_PROCESSOR_OP);
301         break;

303     case AML_DEVICE_OP:
304     case AML_SCOPE_OP:
305     case AML_THERMAL_ZONE_OP:

307         /* Device/Scope/ThermalZone (Namepath) */

309         Length = Op->Asl.FinalAmlLength;
310         NextOp = Op->Asl.Child;     /* Get Namepath */

312         /* Get offset of last nameseg */

314         NamepathOffset = Gbl_CurrentAmlOffset + Length +
315             (NextOp->Asl.FinalAmlLength - ACPI_NAME_SIZE);

317         LsEmitOffsetTableEntry (FileId, Node, NamepathOffset, 0,
318             Op->Asl.ParseOpName, 0, (UINT8) 0, Op->Asl.AmlOpcode);
319         break;

321     default:
322         break;
323     }

325     Gbl_CurrentAmlOffset += Op->Asl.FinalAmlLength;
```



```
326     return (AE_OK);
327 }


330 /*******************************************************************************
331  *
332  * FUNCTION:    LsEmitOffsetTableEntry
333  *
334  * PARAMETERS:  FileId          - ID of current listing file
335  *              Node            - Namespace node associated with the name
336  *              Offset          - Offset of the value within the AML table
337  *              OpName          - Name of the AML opcode
338  *              Value           - Current value of the AML field
339  *              AmlOpcode       - Opcode associated with the field
340  *              ObjectType      - ACPI object type
341  *
342  * RETURN:      None
343  *
344  * DESCRIPTION: Emit a line of the offset table (-so option)
345  *
346  ******************************************************************************/

348 static void
349 LsEmitOffsetTableEntry (
350     UINT32                  FileId,
351     ACPI_NAMESPACE_NODE     *Node,
352     UINT32                  NamepathOffset,
353     UINT32                  Offset,
354     char                    *OpName,
355     UINT64                  Value,
356     UINT8                   AmlOpcode,
357     UINT16                  ParentOpcode)
358 {
359     ACPI_BUFFER             TargetPath;
360     ACPI_STATUS             Status;


363     /* Get the full pathname to the namespace node */

365     TargetPath.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
366     Status = AcpiNsHandleToPathname (Node, &TargetPath);
367     if (ACPI_FAILURE (Status))
368     {
369         return;
370     }

372     /* [1] - Skip the opening backslash for the path */

374     strcpy (MsgBuffer, "\"");
375     strcat (MsgBuffer, &((char *) TargetPath.Pointer)[1]);
376     strcat (MsgBuffer, "\",");
377     ACPI_FREE (TargetPath.Pointer);

379     /*
380      * Max offset is 4G, constrained by 32-bit ACPI table length.
381      * Max Length for Integers is 8 bytes.
382      */
383     FlPrintFile (FileId,
384         "    {%-29s 0x%4.4X, 0x%8.8X, 0x%2.2X, 0x%8.8X, 0x%8.8X%8.8X}, /* %s */\
385         MsgBuffer, ParentOpcode, NamepathOffset, AmlOpcode,
386         Offset, ACPI_FORMAT_UINT64 (Value), OpName);
387 }


390 /*******************************************************************************
391  *
```

```
 * FUNCTION:    LsDoOffsetTableHeader, LsDoOffsetTableFooter
 *
 * PARAMETERS:  FileId          - ID of current listing file
 *
 * RETURN:      None
 *
 * DESCRIPTION: Header and footer for the offset table file.
 *
 *****************************************************************************/

void
LsDoOffsetTableHeader (
    UINT32                  FileId)
{

    FlPrintFile (FileId,
        "#ifndef __AML_OFFSET_TABLE_H\n"
        "#define __AML_OFFSET_TABLE_H\n\n");

    FlPrintFile (FileId, "typedef struct {\n"
        "    char                   *Pathname;      /* Full pathname (from root)
        "    unsigned short         ParentOpcode;   /* AML opcode for the parent
        "    unsigned long          NamesegOffset;  /* Offset of last nameseg in
        "    unsigned char          Opcode;         /* AML opcode for the data *
        "    unsigned long          Offset;         /* Offset for the data */\n"
        "    unsigned long long     Value;          /* Original value of the dat
        "} AML_OFFSET_TABLE_ENTRY;\n\n");

    FlPrintFile (FileId,
        "#endif /* __AML_OFFSET_TABLE_H */\n\n");

    FlPrintFile (FileId,
        "/*\n"
        " * Information specific to the supported object types:\n"
        " *\n"
        " * Integers:\n"
        " *    Opcode is the integer prefix, indicates length of the data\n"
        " *        (One of: BYTE, WORD, DWORD, QWORD, ZERO, ONE, ONES)\n"
        " *    Offset points to the actual integer data\n"
        " *    Value is the existing value in the AML\n"
        " *\n"
        " * Packages:\n"
        " *    Opcode is the package or var_package opcode\n"
        " *    Offset points to the package opcode\n"
        " *    Value is the package element count\n"
        " *\n"
        " * Operation Regions:\n"
        " *    Opcode is the address integer prefix, indicates length of the dat
        " *    Offset points to the region address\n"
        " *    Value is the existing address value in the AML\n"
        " *\n"
        " * Control Methods:\n"
        " *    Offset points to the method flags byte\n"
        " *    Value is the existing flags value in the AML\n"
        " *\n"
        " * Processors:\n"
        " *    Offset points to the first byte of the PBlock Address\n"
        " *\n"
        " * Resource Descriptors:\n"
        " *    Opcode is the descriptor type\n"
        " *    Offset points to the start of the descriptor\n"
        " *\n"
        " * Scopes/Devices/ThermalZones:\n"
        " *    Nameseg offset only\n"
        " */\n");
```



```
    FlPrintFile (FileId,
        "AML_OFFSET_TABLE_ENTRY   %s_%s_OffsetTable[] =\n{\n",
        Gbl_TableSignature, Gbl_TableId);
}


void
LsDoOffsetTableFooter (
    UINT32                  FileId)
{

    FlPrintFile (FileId,
        "    {NULL,0,0,0,0,0} /* Table terminator */\n};\n\n");
    Gbl_CurrentAmlOffset = 0;
}
```

```
**********************************************************
   22710 Thu Dec 26 13:48:31 2013
new/usr/src/common/acpica/compiler/aslopcodes.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslopcode - AML opcode generation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslopcodes")


/* Local prototypes */

static void
OpcDoAccessAs (
    ACPI_PARSE_OBJECT       *Op);

static void
OpcDoConnection (
```



```
    ACPI_PARSE_OBJECT       *Op);

static void
OpcDoUnicode (
    ACPI_PARSE_OBJECT       *Op);

static void
OpcDoEisaId (
    ACPI_PARSE_OBJECT       *Op);

static void
OpcDoUuId (
    ACPI_PARSE_OBJECT       *Op);


/******************************************************************************
 *
 * FUNCTION:    OpcAmlOpcodeUpdateWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Opcode update walk, ascending callback
 *
 *****************************************************************************/

ACPI_STATUS
OpcAmlOpcodeUpdateWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{

    /*
     * Handle the Package() case where the actual opcode cannot be determined
     * until the PackageLength operand has been folded and minimized.
     * (PackageOp versus VarPackageOp)
     *
     * This is (as of ACPI 3.0) the only case where the AML opcode can change
     * based upon the value of a parameter.
     *
     * The parser always inserts a VarPackage opcode, which can possibly be
     * optimized to a Package opcode.
     */
    if (Op->Asl.ParseOpcode == PARSEOP_VAR_PACKAGE)
    {
        OpnDoPackage (Op);
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    OpcAmlOpcodeWalk
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Parse tree walk to generate both the AML opcodes and the AML
 *              operands.
 *
 *****************************************************************************/
```

```
128 ACPI_STATUS
129 OpcAmlOpcodeWalk (
130     ACPI_PARSE_OBJECT       *Op,
131     UINT32                  Level,
132     void                    *Context)
133 {

135     TotalParseNodes++;

137     OpcGenerateAmlOpcode (Op);
138     OpnGenerateAmlOperands (Op);
139     return (AE_OK);
140 }


143 /*******************************************************************************
144  *
145  * FUNCTION:    OpcGetIntegerWidth
146  *
147  * PARAMETERS:  Op          - DEFINITION BLOCK op
148  *
149  * RETURN:      none
150  *
151  * DESCRIPTION: Extract integer width from the table revision
152  *
153  ******************************************************************************/

155 void
156 OpcGetIntegerWidth (
157     ACPI_PARSE_OBJECT       *Op)
158 {
159     ACPI_PARSE_OBJECT       *Child;


162     if (!Op)
163     {
164         return;
165     }

167     if (Gbl_RevisionOverride)
168     {
169         AcpiUtSetIntegerWidth (Gbl_RevisionOverride);
170     }
171     else
172     {
173         Child = Op->Asl.Child;
174         Child = Child->Asl.Next;
175         Child = Child->Asl.Next;

177         /* Use the revision to set the integer width */

179         AcpiUtSetIntegerWidth ((UINT8) Child->Asl.Value.Integer);
180     }
181 }


184 /*******************************************************************************
185  *
186  * FUNCTION:    OpcSetOptimalIntegerSize
187  *
188  * PARAMETERS:  Op        - A parse tree node
189  *
190  * RETURN:      Integer width, in bytes. Also sets the node AML opcode to the
191  *              optimal integer AML prefix opcode.
192  *
```



```
193  * DESCRIPTION: Determine the optimal AML encoding of an integer. All leading
194  *              zeros can be truncated to squeeze the integer into the
195  *              minimal number of AML bytes.
196  *
197  ******************************************************************************/

199 UINT32
200 OpcSetOptimalIntegerSize (
201     ACPI_PARSE_OBJECT       *Op)
202 {

204 #if 0
205     /*
206      * TBD: - we don't want to optimize integers in the block header, but the
207      * code below does not work correctly.
208      */
209     if (Op->Asl.Parent &&
210         Op->Asl.Parent->Asl.Parent &&
211        (Op->Asl.Parent->Asl.Parent->Asl.ParseOpcode == PARSEOP_DEFINITIONBLOCK))
212     {
213         return (0);
214     }
215 #endif

217     /*
218      * Check for the special AML integers first - Zero, One, Ones.
219      * These are single-byte opcodes that are the smallest possible
220      * representation of an integer.
221      *
222      * This optimization is optional.
223      */
224     if (Gbl_IntegerOptimizationFlag)
225     {
226         switch (Op->Asl.Value.Integer)
227         {
228         case 0:

230             Op->Asl.AmlOpcode = AML_ZERO_OP;
231             AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
232                 Op, "Zero");
233             return (1);

235         case 1:

237             Op->Asl.AmlOpcode = AML_ONE_OP;
238             AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
239                 Op, "One");
240             return (1);

242         case ACPI_UINT32_MAX:

244             /* Check for table integer width (32 or 64) */

246             if (AcpiGbl_IntegerByteWidth == 4)
247             {
248                 Op->Asl.AmlOpcode = AML_ONES_OP;
249                 AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
250                     Op, "Ones");
251                 return (1);
252             }
253             break;

255         case ACPI_UINT64_MAX:

257             /* Check for table integer width (32 or 64) */
```

```
259             if (AcpiGbl_IntegerByteWidth == 8)
260             {
261                 Op->Asl.AmlOpcode = AML_ONES_OP;
262                 AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
263                     Op, "Ones");
264                 return (1);
265             }
266             break;

268         default:

270             break;
271         }
272     }

274     /* Find the best fit using the various AML integer prefixes */

276     if (Op->Asl.Value.Integer <= ACPI_UINT8_MAX)
277     {
278         Op->Asl.AmlOpcode = AML_BYTE_OP;
279         return (1);
280     }
281     if (Op->Asl.Value.Integer <= ACPI_UINT16_MAX)
282     {
283         Op->Asl.AmlOpcode = AML_WORD_OP;
284         return (2);
285     }
286     if (Op->Asl.Value.Integer <= ACPI_UINT32_MAX)
287     {
288         Op->Asl.AmlOpcode = AML_DWORD_OP;
289         return (4);
290     }
291     else
292     {
293         if (AcpiGbl_IntegerByteWidth == 4)
294         {
295             AslError (ASL_WARNING, ASL_MSG_INTEGER_LENGTH,
296                 Op, NULL);

298             if (!Gbl_IgnoreErrors)
299             {
300                 /* Truncate the integer to 32-bit */
301                 Op->Asl.AmlOpcode = AML_DWORD_OP;
302                 return (4);
303             }
304         }

306         Op->Asl.AmlOpcode = AML_QWORD_OP;
307         return (8);
308     }
309 }


312 /*******************************************************************************
313  *
314  * FUNCTION:    OpcDoAccessAs
315  *
316  * PARAMETERS:  Op        - Parse node
317  *
318  * RETURN:      None
319  *
320  * DESCRIPTION: Implement the ACCESS_AS ASL keyword.
321  *
322  ******************************************************************************/

324 static void
```



```
325 OpcDoAccessAs (
326     ACPI_PARSE_OBJECT       *Op)
327 {
328     ACPI_PARSE_OBJECT       *TypeOp;
329     ACPI_PARSE_OBJECT       *AttribOp;
330     ACPI_PARSE_OBJECT       *LengthOp;
331     UINT8                   Attribute;


334     Op->Asl.AmlOpcodeLength = 1;
335     TypeOp = Op->Asl.Child;

337     /* First child is the access type */

339     TypeOp->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
340     TypeOp->Asl.ParseOpcode = PARSEOP_RAW_DATA;

342     /* Second child is the optional access attribute */

344     AttribOp = TypeOp->Asl.Next;
345     if (AttribOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
346     {
347         AttribOp->Asl.Value.Integer = 0;
348     }
349     AttribOp->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
350     AttribOp->Asl.ParseOpcode = PARSEOP_RAW_DATA;

352     /* Only a few AccessAttributes support AccessLength */

354     Attribute = (UINT8) AttribOp->Asl.Value.Integer;
355     if ((Attribute != AML_FIELD_ATTRIB_MULTIBYTE) &&
356         (Attribute != AML_FIELD_ATTRIB_RAW_BYTES) &&
357         (Attribute != AML_FIELD_ATTRIB_RAW_PROCESS))
358     {
359         return;
360     }

362     Op->Asl.AmlOpcode = AML_FIELD_EXT_ACCESS_OP;

364     /*
365      * Child of Attributes is the AccessLength (required for Multibyte,
366      * RawBytes, RawProcess.)
367      */
368     LengthOp = AttribOp->Asl.Child;
369     if (!LengthOp)
370     {
371         return;
372     }

374     /* TBD: probably can remove */

376     if (LengthOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
377     {
378         LengthOp->Asl.Value.Integer = 16;
379     }

381     LengthOp->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
382     LengthOp->Asl.ParseOpcode = PARSEOP_RAW_DATA;
383 }


386 /*******************************************************************************
387  *
388  * FUNCTION:    OpcDoConnection
389  *
390  * PARAMETERS:  Op        - Parse node
```

```
 *
 * RETURN:      None
 *
 * DESCRIPTION: Implement the Connection ASL keyword.
 *
 *****************************************************************************/

static void
OpcDoConnection (
    ACPI_PARSE_OBJECT       *Op)
{
    ASL_RESOURCE_NODE       *Rnode;
    ACPI_PARSE_OBJECT       *BufferOp;
    ACPI_PARSE_OBJECT       *BufferLengthOp;
    ACPI_PARSE_OBJECT       *BufferDataOp;
    UINT8                   State;


    Op->Asl.AmlOpcodeLength = 1;

    if (Op->Asl.Child->Asl.AmlOpcode == AML_INT_NAMEPATH_OP)
    {
        return;
    }

    BufferOp = Op->Asl.Child;
    BufferLengthOp = BufferOp->Asl.Child;
    BufferDataOp = BufferLengthOp->Asl.Next;

    State = ACPI_RSTATE_NORMAL;
    Rnode = RsDoOneResourceDescriptor (BufferDataOp->Asl.Next, 0, &State);
    if (!Rnode)
    {
        return; /* error */
    }

    /*
     * Transform the nodes into the following
     *
     * Op           -> AML_BUFFER_OP
     * First Child  -> BufferLength
     * Second Child -> Descriptor Buffer (raw byte data)
     */
    BufferOp->Asl.ParseOpcode         = PARSEOP_BUFFER;
    BufferOp->Asl.AmlOpcode           = AML_BUFFER_OP;
    BufferOp->Asl.CompileFlags        = NODE_AML_PACKAGE | NODE_IS_RESOURCE_DESC
    UtSetParseOpName (BufferOp);

    BufferLengthOp->Asl.ParseOpcode   = PARSEOP_INTEGER;
    BufferLengthOp->Asl.Value.Integer = Rnode->BufferLength;
    (void) OpcSetOptimalIntegerSize (BufferLengthOp);
    UtSetParseOpName (BufferLengthOp);

    BufferDataOp->Asl.ParseOpcode         = PARSEOP_RAW_DATA;
    BufferDataOp->Asl.AmlOpcode           = AML_RAW_DATA_CHAIN;
    BufferDataOp->Asl.AmlOpcodeLength     = 0;
    BufferDataOp->Asl.AmlLength           = Rnode->BufferLength;
    BufferDataOp->Asl.Value.Buffer        = (UINT8 *) Rnode;
    UtSetParseOpName (BufferDataOp);
}


/*****************************************************************************
 *
 * FUNCTION:    OpcDoUnicode
 *
```



```
 * PARAMETERS:  Op        - Parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Implement the UNICODE ASL "macro".  Convert the input string
 *              to a unicode buffer. There is no Unicode AML opcode.
 *
 * Note:  The Unicode string is 16 bits per character, no leading signature,
 *        with a 16-bit terminating NULL.
 *
 *****************************************************************************/

static void
OpcDoUnicode (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *InitializerOp;
    UINT32                  Length;
    UINT32                  Count;
    UINT32                  i;
    UINT8                   *AsciiString;
    UINT16                  *UnicodeString;
    ACPI_PARSE_OBJECT       *BufferLengthOp;


    /* Change op into a buffer object */

    Op->Asl.CompileFlags &= ~NODE_COMPILE_TIME_CONST;
    Op->Asl.ParseOpcode = PARSEOP_BUFFER;
    UtSetParseOpName (Op);

    /* Buffer Length is first, followed by the string */

    BufferLengthOp = Op->Asl.Child;
    InitializerOp = BufferLengthOp->Asl.Next;

    AsciiString = (UINT8 *) InitializerOp->Asl.Value.String;

    /* Create a new buffer for the Unicode string */

    Count = strlen (InitializerOp->Asl.Value.String) + 1;
    Length = Count * sizeof (UINT16);
    UnicodeString = UtLocalCalloc (Length);

    /* Convert to Unicode string (including null terminator) */

    for (i = 0; i < Count; i++)
    {
        UnicodeString[i] = (UINT16) AsciiString[i];
    }

    /*
     * Just set the buffer size node to be the buffer length, regardless
     * of whether it was previously an integer or a default_arg placeholder
     */
    BufferLengthOp->Asl.ParseOpcode   = PARSEOP_INTEGER;
    BufferLengthOp->Asl.AmlOpcode     = AML_DWORD_OP;
    BufferLengthOp->Asl.Value.Integer = Length;
    UtSetParseOpName (BufferLengthOp);

    (void) OpcSetOptimalIntegerSize (BufferLengthOp);

    /* The Unicode string is a raw data buffer */

    InitializerOp->Asl.Value.Buffer   = (UINT8 *) UnicodeString;
    InitializerOp->Asl.AmlOpcode      = AML_RAW_DATA_BUFFER;
```

```
523     InitializerOp->Asl.AmlLength      = Length;
524     InitializerOp->Asl.ParseOpcode    = PARSEOP_RAW_DATA;
525     InitializerOp->Asl.Child          = NULL;
526     UtSetParseOpName (InitializerOp);
527 }


530 /*******************************************************************************
531  *
532  * FUNCTION:    OpcDoEisaId
533  *
534  * PARAMETERS:  Op        - Parse node
535  *
536  * RETURN:      None
537  *
538  * DESCRIPTION: Convert a string EISA ID to numeric representation. See the
539  *              Pnp BIOS Specification for details. Here is an excerpt:
540  *
541  *              A seven character ASCII representation of the product
542  *              identifier compressed into a 32-bit identifier. The seven
543  *              character ID consists of a three character manufacturer code,
544  *              a three character hexadecimal product identifier, and a one
545  *              character hexadecimal revision number. The manufacturer code
546  *              is a 3 uppercase character code that is compressed into 3 5-bit
547  *              values as follows:
548  *                  1) Find hex ASCII value for each letter
549  *                  2) Subtract 40h from each ASCII value
550  *                  3) Retain 5 least significant bits for each letter by
551  *                     discarding upper 3 bits because they are always 0.
552  *                  4) Compressed code = concatenate 0 and the 3 5-bit values
553  *
554  *              The format of the compressed product identifier is as follows:
555  *              Byte 0: Bit 7       - Reserved (0)
556  *                      Bits 6-2:   - 1st character of compressed mfg code
557  *                      Bits 1-0    - Upper 2 bits of 2nd character of mfg code
558  *              Byte 1: Bits 7-5    - Lower 3 bits of 2nd character of mfg code
559  *                      Bits 4-0    - 3rd character of mfg code
560  *              Byte 2: Bits 7-4    - 1st hex digit of product number
561  *                      Bits 3-0    - 2nd hex digit of product number
562  *              Byte 3: Bits 7-4    - 3st hex digit of product number
563  *                      Bits 3-0    - Hex digit of the revision number
564  *
565  ******************************************************************************/

567 static void
568 OpcDoEisaId (
569     ACPI_PARSE_OBJECT       *Op)
570 {
571     UINT32                  EisaId = 0;
572     UINT32                  BigEndianId;
573     char                    *InString;
574     ACPI_STATUS             Status = AE_OK;
575     UINT32                  i;


578     InString = (char *) Op->Asl.Value.String;

580     /*
581      * The EISAID string must be exactly 7 characters and of the form
582      * "UUUXXXX" -- 3 uppercase letters and 4 hex digits (e.g., "PNP0001")
583      */
584     if (ACPI_STRLEN (InString) != 7)
585     {
586         Status = AE_BAD_PARAMETER;
587     }
588     else
```



```
589     {
590         /* Check all 7 characters for correct format */

592         for (i = 0; i < 7; i++)
593         {
594             /* First 3 characters must be uppercase letters */

596             if (i < 3)
597             {
598                 if (!isupper ((int) InString[i]))
599                 {
600                     Status = AE_BAD_PARAMETER;
601                 }
602             }

604             /* Last 4 characters must be hex digits */

606             else if (!isxdigit ((int) InString[i]))
607             {
608                 Status = AE_BAD_PARAMETER;
609             }
610         }
611     }

613     if (ACPI_FAILURE (Status))
614     {
615         AslError (ASL_ERROR, ASL_MSG_INVALID_EISAID, Op, Op->Asl.Value.String);
616     }
617     else
618     {
619         /* Create ID big-endian first (bits are contiguous) */

621         BigEndianId =
622             (UINT32) ((UINT8) (InString[0] - 0x40)) << 26 |
623             (UINT32) ((UINT8) (InString[1] - 0x40)) << 21 |
624             (UINT32) ((UINT8) (InString[2] - 0x40)) << 16 |

626             (UtHexCharToValue (InString[3])) << 12 |
627             (UtHexCharToValue (InString[4])) << 8  |
628             (UtHexCharToValue (InString[5])) << 4  |
629              UtHexCharToValue (InString[6]);

631         /* Swap to little-endian to get final ID (see function header) */

633         EisaId = AcpiUtDwordByteSwap (BigEndianId);
634     }

636     /*
637      * Morph the Op into an integer, regardless of whether there
638      * was an error in the EISAID string
639      */
640     Op->Asl.Value.Integer = EisaId;

642     Op->Asl.CompileFlags &= ~NODE_COMPILE_TIME_CONST;
643     Op->Asl.ParseOpcode = PARSEOP_INTEGER;
644     (void) OpcSetOptimalIntegerSize (Op);

646     /* Op is now an integer */

648     UtSetParseOpName (Op);
649 }


652 /*******************************************************************************
653  *
654  * FUNCTION:    OpcDoUuId
```

```
 *
 * PARAMETERS:  Op        - Parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Convert UUID string to 16-byte buffer
 *
 *****************************************************************************/

static void
OpcDoUuId (
    ACPI_PARSE_OBJECT       *Op)
{
    char                    *InString;
    char                    *Buffer;
    ACPI_STATUS             Status = AE_OK;
    ACPI_PARSE_OBJECT       *NewOp;


    InString = (char *) Op->Asl.Value.String;
    Buffer = UtLocalCalloc (16);

    Status = AuValidateUuid (InString);
    if (ACPI_FAILURE (Status))
    {
        AslError (ASL_ERROR, ASL_MSG_INVALID_UUID, Op, Op->Asl.Value.String);
    }
    else
    {
        (void) AuConvertStringToUuid (InString, Buffer);
    }

    /* Change Op to a Buffer */

    Op->Asl.ParseOpcode = PARSEOP_BUFFER;
    Op->Common.AmlOpcode = AML_BUFFER_OP;

    /* Disable further optimization */

    Op->Asl.CompileFlags &= ~NODE_COMPILE_TIME_CONST;
    UtSetParseOpName (Op);

    /* Child node is the buffer length */

    NewOp = TrAllocateNode (PARSEOP_INTEGER);

    NewOp->Asl.AmlOpcode     = AML_BYTE_OP;
    NewOp->Asl.Value.Integer = 16;
    NewOp->Asl.Parent        = Op;

    Op->Asl.Child = NewOp;
    Op = NewOp;

    /* Peer to the child is the raw buffer data */

    NewOp = TrAllocateNode (PARSEOP_RAW_DATA);
    NewOp->Asl.AmlOpcode     = AML_RAW_DATA_BUFFER;
    NewOp->Asl.AmlLength     = 16;
    NewOp->Asl.Value.String  = (char *) Buffer;
    NewOp->Asl.Parent        = Op->Asl.Parent;

    Op->Asl.Next = NewOp;
}


/*****************************************************************************
```



```
 *
 * FUNCTION:    OpcGenerateAmlOpcode
 *
 * PARAMETERS:  Op        - Parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Generate the AML opcode associated with the node and its
 *              parse (lex/flex) keyword opcode. Essentially implements
 *              a mapping between the parse opcodes and the actual AML opcodes.
 *
 *****************************************************************************/

void
OpcGenerateAmlOpcode (
    ACPI_PARSE_OBJECT       *Op)
{

    UINT16                  Index;


    Index = (UINT16) (Op->Asl.ParseOpcode - ASL_PARSE_OPCODE_BASE);

    Op->Asl.AmlOpcode     = AslKeywordMapping[Index].AmlOpcode;
    Op->Asl.AcpiBtype     = AslKeywordMapping[Index].AcpiBtype;
    Op->Asl.CompileFlags |= AslKeywordMapping[Index].Flags;

    if (!Op->Asl.Value.Integer)
    {
        Op->Asl.Value.Integer = AslKeywordMapping[Index].Value;
    }

    /* Special handling for some opcodes */

    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_INTEGER:
        /*
         * Set the opcode based on the size of the integer
         */
        (void) OpcSetOptimalIntegerSize (Op);
        break;

    case PARSEOP_OFFSET:

        Op->Asl.AmlOpcodeLength = 1;
        break;

    case PARSEOP_ACCESSAS:

        OpcDoAccessAs (Op);
        break;

    case PARSEOP_CONNECTION:

        OpcDoConnection (Op);
        break;

    case PARSEOP_EISAID:

        OpcDoEisaId (Op);
        break;

    case PARSEOP_TOUUID:

        OpcDoUuId (Op);
```

```
787         break;
789     case PARSEOP_UNICODE:
791         OpcDoUnicode (Op);
792         break;
794     case PARSEOP_INCLUDE:
796         Op->Asl.Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
797         Gbl_HasIncludeFiles = TRUE;
798         break;
800     case PARSEOP_EXTERNAL:
802         Op->Asl.Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
803         Op->Asl.Child->Asl.Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
804         break;
806     case PARSEOP_TIMER:
808         if (AcpiGbl_IntegerBitWidth == 32)
809         {
810             AslError (ASL_REMARK, ASL_MSG_TRUNCATION, Op, NULL);
811         }
812         break;
814     default:
816         /* Nothing to do for other opcodes */
818         break;
819     }
821     return;
822 }
```

```
**********************************************************
   31757 Thu Dec 26 13:48:31 2013
new/usr/src/common/acpica/compiler/asloperands.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asloperands - AML operand processing
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asloperands")

/* Local prototypes */

static void
OpnDoField (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoBankField (
```



```
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoBuffer (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoDefinitionBlock (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoFieldCommon (
    ACPI_PARSE_OBJECT       *FieldOp,
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoIndexField (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoLoadTable (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoMethod (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoMutex (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnDoRegion (
    ACPI_PARSE_OBJECT       *Op);

static void
OpnAttachNameToNode (
    ACPI_PARSE_OBJECT       *Op);


/******************************************************************************
 *
 * FUNCTION:    OpnDoMutex
 *
 * PARAMETERS:  Op        - The parent parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Construct the operands for the MUTEX ASL keyword.
 *
 *****************************************************************************/

static void
OpnDoMutex (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Next;


    Next = Op->Asl.Child;
    Next = Next->Asl.Next;

    if (Next->Asl.Value.Integer > 15)
    {
        AslError (ASL_ERROR, ASL_MSG_SYNC_LEVEL, Next, NULL);
    }
```

```
126     return;
127 }


130 /*******************************************************************************
131  *
132  * FUNCTION:    OpnDoMethod
133  *
134  * PARAMETERS:  Op        - The parent parse node
135  *
136  * RETURN:      None
137  *
138  * DESCRIPTION: Construct the operands for the METHOD ASL keyword.
139  *
140  ******************************************************************************/

142 static void
143 OpnDoMethod (
144     ACPI_PARSE_OBJECT       *Op)
145 {
146     ACPI_PARSE_OBJECT       *Next;

148     /* Optional arguments for this opcode with defaults */

150     UINT8                   NumArgs = 0;
151     UINT8                   Serialized = 0;
152     UINT8                   Concurrency = 0;
153     UINT8                   MethodFlags;


156     /* Opcode and package length first */
157     /* Method name */

159     Next = Op->Asl.Child;

161     /* Num args */

163     Next = Next->Asl.Next;
164     if (Next->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
165     {
166         NumArgs = (UINT8) Next->Asl.Value.Integer;
167         Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
168     }

170     /* Serialized Flag */

172     Next = Next->Asl.Next;
173     if (Next->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
174     {
175         Serialized = (UINT8) Next->Asl.Value.Integer;
176         Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
177     }

179     /* Concurrency value (valid values are 0-15) */

181     Next = Next->Asl.Next;
182     if (Next->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
183     {
184         /* This is a ByteConstExpr, so eval the constant now */

186         OpcAmlConstantWalk (Next, 0, NULL);

188         if (Next->Asl.Value.Integer > 15)
189         {
190             AslError (ASL_ERROR, ASL_MSG_SYNC_LEVEL, Next, NULL);
191         }
```



```
192         Concurrency = (UINT8) Next->Asl.Value.Integer;
193     }

195     /* Put the bits in their proper places */

197     MethodFlags = (UINT8) ((NumArgs & 0x7) |
198                            ((Serialized & 0x1) << 3) |
199                            ((Concurrency & 0xF) << 4));

201     /* Use the last node for the combined flags byte */

203     Next->Asl.Value.Integer = MethodFlags;
204     Next->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
205     Next->Asl.AmlLength = 1;
206     Next->Asl.ParseOpcode = PARSEOP_RAW_DATA;

208     /* Save the arg count in the first node */

210     Op->Asl.Extra = NumArgs;
211 }


214 /*******************************************************************************
215  *
216  * FUNCTION:    OpnDoFieldCommon
217  *
218  * PARAMETERS:  FieldOp       - Node for an ASL field
219  *              Op            - The parent parse node
220  *
221  * RETURN:      None
222  *
223  * DESCRIPTION: Construct the AML operands for the various field keywords,
224  *              FIELD, BANKFIELD, INDEXFIELD
225  *
226  ******************************************************************************/

228 static void
229 OpnDoFieldCommon (
230     ACPI_PARSE_OBJECT       *FieldOp,
231     ACPI_PARSE_OBJECT       *Op)
232 {
233     ACPI_PARSE_OBJECT       *Next;
234     ACPI_PARSE_OBJECT       *PkgLengthNode;
235     UINT32                  CurrentBitOffset;
236     UINT32                  NewBitOffset;
237     UINT8                   AccessType;
238     UINT8                   LockRule;
239     UINT8                   UpdateRule;
240     UINT8                   FieldFlags;
241     UINT32                  MinimumLength;


244     /* AccessType -- not optional, so no need to check for DEFAULT_ARG */

246     AccessType = (UINT8) Op->Asl.Value.Integer;
247     Op->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;

249     /* Set the access type in the parent (field) node for use later */

251     FieldOp->Asl.Value.Integer = AccessType;

253     /* LockRule -- not optional, so no need to check for DEFAULT_ARG */

255     Next = Op->Asl.Next;
256     LockRule = (UINT8) Next->Asl.Value.Integer;
257     Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
```

```
259     /* UpdateRule -- not optional, so no need to check for DEFAULT_ARG */

261     Next = Next->Asl.Next;
262     UpdateRule = (UINT8) Next->Asl.Value.Integer;

264     /*
265      * Generate the flags byte. The various fields are already
266      * in the right bit position via translation from the
267      * keywords by the parser.
268      */
269     FieldFlags = (UINT8) (AccessType | LockRule | UpdateRule);

271     /* Use the previous node to be the FieldFlags node */

273     /* Set the node to RAW_DATA */

275     Next->Asl.Value.Integer = FieldFlags;
276     Next->Asl.AmlOpcode     = AML_RAW_DATA_BYTE;
277     Next->Asl.AmlLength     = 1;
278     Next->Asl.ParseOpcode   = PARSEOP_RAW_DATA;

280     /* Process the FieldUnitList */

282     Next = Next->Asl.Next;
283     CurrentBitOffset = 0;

285     while (Next)
286     {
287         /* Save the offset of this field unit */

289         Next->Asl.ExtraValue = CurrentBitOffset;

291         switch (Next->Asl.ParseOpcode)
292         {
293         case PARSEOP_ACCESSAS:

295             PkgLengthNode = Next->Asl.Child;
296             AccessType = (UINT8) PkgLengthNode->Asl.Value.Integer;

298             /* Nothing additional to do */
299             break;

301         case PARSEOP_OFFSET:

303             /* New offset into the field */

305             PkgLengthNode = Next->Asl.Child;
306             NewBitOffset = ((UINT32) PkgLengthNode->Asl.Value.Integer) * 8;

308             /*
309              * Examine the specified offset in relation to the
310              * current offset counter.
311              */
312             if (NewBitOffset < CurrentBitOffset)
313             {
314                 /*
315                  * Not allowed to specify a backwards offset!
316                  * Issue error and ignore this node.
317                  */
318                 AslError (ASL_ERROR, ASL_MSG_BACKWARDS_OFFSET, PkgLengthNode,
319                     NULL);
320                 Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
321                 PkgLengthNode->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
322             }
323             else if (NewBitOffset == CurrentBitOffset)
```



```
324             {
325                 /*
326                  * Offset is redundant; we don't need to output an
327                  * offset opcode. Just set these nodes to default
328                  */
329                 Next->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
330                 PkgLengthNode->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
331             }
332             else
333             {
334                 /*
335                  * Valid new offset - set the value to be inserted into the AML
336                  * and update the offset counter.
337                  */
338                 PkgLengthNode->Asl.Value.Integer =
339                     NewBitOffset - CurrentBitOffset;
340                 CurrentBitOffset = NewBitOffset;
341             }
342             break;

344         case PARSEOP_NAMESEG:
345         case PARSEOP_RESERVED_BYTES:

347             /* Named or reserved field entry */

349             PkgLengthNode     = Next->Asl.Child;
350             NewBitOffset      = (UINT32) PkgLengthNode->Asl.Value.Integer;
351             CurrentBitOffset += NewBitOffset;

353             /* Save the current AccessAs value for error checking later */

355             switch (AccessType)
356             {
357                 case AML_FIELD_ACCESS_ANY:
358                 case AML_FIELD_ACCESS_BYTE:
359                 case AML_FIELD_ACCESS_BUFFER:
360                 default:

362                     MinimumLength = 8;
363                     break;

365                 case AML_FIELD_ACCESS_WORD:
366                     MinimumLength = 16;
367                     break;

369                 case AML_FIELD_ACCESS_DWORD:
370                     MinimumLength = 32;
371                     break;

373                 case AML_FIELD_ACCESS_QWORD:
374                     MinimumLength = 64;
375                     break;
376             }

378             PkgLengthNode->Asl.ExtraValue = MinimumLength;
379             break;

381         default:

383             /* All supported field opcodes must appear above */

385             break;
386         }

388         /* Move on to next entry in the field list */
```

```
390         Next = Next->Asl.Next;
391     }
392 }


395 /*******************************************************************************
396  *
397  * FUNCTION:    OpnDoField
398  *
399  * PARAMETERS:  Op        - The parent parse node
400  *
401  * RETURN:      None
402  *
403  * DESCRIPTION: Construct the AML operands for the FIELD ASL keyword
404  *
405  ******************************************************************************/

407 static void
408 OpnDoField (
409     ACPI_PARSE_OBJECT       *Op)
410 {
411     ACPI_PARSE_OBJECT       *Next;


414     /* Opcode is parent node */
415     /* First child is field name */

417     Next = Op->Asl.Child;

419     /* Second child is the AccessType */

421     OpnDoFieldCommon (Op, Next->Asl.Next);
422 }


425 /*******************************************************************************
426  *
427  * FUNCTION:    OpnDoIndexField
428  *
429  * PARAMETERS:  Op        - The parent parse node
430  *
431  * RETURN:      None
432  *
433  * DESCRIPTION: Construct the AML operands for the INDEXFIELD ASL keyword
434  *
435  ******************************************************************************/

437 static void
438 OpnDoIndexField (
439     ACPI_PARSE_OBJECT       *Op)
440 {
441     ACPI_PARSE_OBJECT       *Next;


444     /* Opcode is parent node */
445     /* First child is the index name */

447     Next = Op->Asl.Child;

449     /* Second child is the data name */

451     Next = Next->Asl.Next;

453     /* Third child is the AccessType */

455     OpnDoFieldCommon (Op, Next->Asl.Next);
```



```
456 }


459 /*******************************************************************************
460  *
461  * FUNCTION:    OpnDoBankField
462  *
463  * PARAMETERS:  Op        - The parent parse node
464  *
465  * RETURN:      None
466  *
467  * DESCRIPTION: Construct the AML operands for the BANKFIELD ASL keyword
468  *
469  ******************************************************************************/

471 static void
472 OpnDoBankField (
473     ACPI_PARSE_OBJECT       *Op)
474 {
475     ACPI_PARSE_OBJECT       *Next;


478     /* Opcode is parent node */
479     /* First child is the region name */

481     Next = Op->Asl.Child;

483     /* Second child is the bank name */

485     Next = Next->Asl.Next;

487     /* Third child is the bank value */

489     Next = Next->Asl.Next;

491     /* Fourth child is the AccessType */

493     OpnDoFieldCommon (Op, Next->Asl.Next);
494 }


497 /*******************************************************************************
498  *
499  * FUNCTION:    OpnDoRegion
500  *
501  * PARAMETERS:  Op        - The parent parse node
502  *
503  * RETURN:      None
504  *
505  * DESCRIPTION: Tries to get the length of the region. Can only do this at
506  *              compile time if the length is a constant.
507  *
508  ******************************************************************************/

510 static void
511 OpnDoRegion (
512     ACPI_PARSE_OBJECT       *Op)
513 {
514     ACPI_PARSE_OBJECT       *Next;


517     /* Opcode is parent node */
518     /* First child is the region name */

520     Next = Op->Asl.Child;
```

```
522     /* Second child is the space ID*/

524     Next = Next->Asl.Next;

526     /* Third child is the region offset */

528     Next = Next->Asl.Next;

530     /* Fourth child is the region length */

532     Next = Next->Asl.Next;
533     if (Next->Asl.ParseOpcode == PARSEOP_INTEGER)
534     {
535         Op->Asl.Value.Integer = Next->Asl.Value.Integer;
536     }
537     else
538     {
539         Op->Asl.Value.Integer = ACPI_UINT64_MAX;
540     }
541 }


544 /*******************************************************************************
545  *
546  * FUNCTION:    OpnDoBuffer
547  *
548  * PARAMETERS:  Op        - The parent parse node
549  *
550  * RETURN:      None
551  *
552  * DESCRIPTION: Construct the AML operands for the BUFFER ASL keyword. We
553  *              build a single raw byte buffer from the initialization nodes,
554  *              each parse node contains a buffer byte.
555  *
556  ******************************************************************************/

558 static void
559 OpnDoBuffer (
560     ACPI_PARSE_OBJECT       *Op)
561 {
562     ACPI_PARSE_OBJECT       *InitializerOp;
563     ACPI_PARSE_OBJECT       *BufferLengthOp;

565     /* Optional arguments for this opcode with defaults */

567     UINT32                  BufferLength = 0;


570     /* Opcode and package length first */
571     /* Buffer Length is next, followed by the initializer list */

573     BufferLengthOp = Op->Asl.Child;
574     InitializerOp = BufferLengthOp->Asl.Next;

576     /*
577      * If the BufferLength is not an INTEGER or was not specified in the ASL
578      * (DEFAULT_ARG), it is a TermArg that is
579      * evaluated at run-time, and we are therefore finished.
580      */
581     if ((BufferLengthOp->Asl.ParseOpcode != PARSEOP_INTEGER) &&
582         (BufferLengthOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG))
583     {
584         return;
585     }

587     /*
```



```
588      * We want to count the number of items in the initializer list, because if
589      * it is larger than the buffer length, we will define the buffer size
590      * to be the size of the initializer list (as per the ACPI Specification)
591      */
592     switch (InitializerOp->Asl.ParseOpcode)
593     {
594     case PARSEOP_INTEGER:
595     case PARSEOP_BYTECONST:
596     case PARSEOP_WORDCONST:
597     case PARSEOP_DWORDCONST:

599         /* The peer list contains the byte list (if any...) */

601         while (InitializerOp)
602         {
603             /* For buffers, this is a list of raw bytes */

605             InitializerOp->Asl.AmlOpcode      = AML_RAW_DATA_BYTE;
606             InitializerOp->Asl.AmlLength      = 1;
607             InitializerOp->Asl.ParseOpcode    = PARSEOP_RAW_DATA;

609             BufferLength++;
610             InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
611         }
612         break;

614     case PARSEOP_STRING_LITERAL:

616         /*
617          * Only one initializer, the string. Buffer must be big enough to hold
618          * the string plus the null termination byte
619          */
620         BufferLength = strlen (InitializerOp->Asl.Value.String) + 1;

622         InitializerOp->Asl.AmlOpcode      = AML_RAW_DATA_BUFFER;
623         InitializerOp->Asl.AmlLength      = BufferLength;
624         InitializerOp->Asl.ParseOpcode    = PARSEOP_RAW_DATA;
625         break;

627     case PARSEOP_RAW_DATA:

629         /* Buffer nodes are already initialized (e.g. Unicode operator) */
630         return;

632     case PARSEOP_DEFAULT_ARG:
633         break;

635     default:

637         AslError (ASL_ERROR, ASL_MSG_INVALID_OPERAND, InitializerOp,
638             "Unknown buffer initializer opcode");
639         printf ("Unknown buffer initializer opcode [%s]\n",
640                         UtGetOpName (InitializerOp->Asl.ParseOpcode));
641         return;
642     }

644     /* Check if initializer list is longer than the buffer length */

646     if (BufferLengthOp->Asl.Value.Integer > BufferLength)
647     {
648         BufferLength = (UINT32) BufferLengthOp->Asl.Value.Integer;
649     }

651     if (!BufferLength)
652     {
653         /* No length AND no items -- issue notice */
```

```
655         AslError (ASL_REMARK, ASL_MSG_BUFFER_LENGTH, BufferLengthOp, NULL);

657         /* But go ahead and put the buffer length of zero into the AML */
658     }

660     /*
661      * Just set the buffer size node to be the buffer length, regardless
662      * of whether it was previously an integer or a default_arg placeholder
663      */
664     BufferLengthOp->Asl.ParseOpcode   = PARSEOP_INTEGER;
665     BufferLengthOp->Asl.AmlOpcode     = AML_DWORD_OP;
666     BufferLengthOp->Asl.Value.Integer = BufferLength;

668     (void) OpcSetOptimalIntegerSize (BufferLengthOp);

670     /* Remaining nodes are handled via the tree walk */
671 }


674 /*******************************************************************************
675  *
676  * FUNCTION:    OpnDoPackage
677  *
678  * PARAMETERS:  Op        - The parent parse node
679  *
680  * RETURN:      None
681  *
682  * DESCRIPTION: Construct the AML operands for the PACKAGE ASL keyword. NOTE:
683  *              can only be called after constants have been folded, to ensure
684  *              that the PackageLength operand has been fully reduced.
685  *
686  ******************************************************************************/

688 void
689 OpnDoPackage (
690     ACPI_PARSE_OBJECT       *Op)
691 {
692     ACPI_PARSE_OBJECT       *InitializerOp;
693     ACPI_PARSE_OBJECT       *PackageLengthOp;
694     UINT32                  PackageLength = 0;


697     /* Opcode and package length first, followed by the initializer list */

699     PackageLengthOp = Op->Asl.Child;
700     InitializerOp = PackageLengthOp->Asl.Next;

702     /* Count the number of items in the initializer list */

704     if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
705     {
706         /* The peer list contains the byte list (if any...) */

708         while (InitializerOp)
709         {
710             PackageLength++;
711             InitializerOp = InitializerOp->Asl.Next;
712         }
713     }

715     /* If package length is a constant, compare to the initializer list */

717     if ((PackageLengthOp->Asl.ParseOpcode == PARSEOP_INTEGER)      ||
718         (PackageLengthOp->Asl.ParseOpcode == PARSEOP_QWORDCONST))
719     {
```



```
720         if (PackageLengthOp->Asl.Value.Integer > PackageLength)
721         {
722             /*
723              * Allow package length to be longer than the initializer
724              * list -- but if the length of initializer list is nonzero,
725              * issue a message since this is probably a coding error,
726              * even though technically legal.
727              */
728             if (PackageLength > 0)
729             {
730                 AslError (ASL_REMARK, ASL_MSG_LIST_LENGTH_SHORT,
731                     PackageLengthOp, NULL);
732             }

734             PackageLength = (UINT32) PackageLengthOp->Asl.Value.Integer;
735         }
736         else if (PackageLengthOp->Asl.Value.Integer < PackageLength)
737         {
738             /*
739              * The package length is smaller than the length of the
740              * initializer list. This is an error as per the ACPI spec.
741              */
742             AslError (ASL_ERROR, ASL_MSG_LIST_LENGTH_LONG,
743                 PackageLengthOp, NULL);
744         }
745     }

747     if (PackageLengthOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
748     {
749         /*
750          * This is the case if the PackageLength was left empty - Package()
751          * The package length becomes the length of the initializer list
752          */
753         Op->Asl.Child->Asl.ParseOpcode = PARSEOP_INTEGER;
754         Op->Asl.Child->Asl.Value.Integer = PackageLength;

756         /* Set the AML opcode */

758         (void) OpcSetOptimalIntegerSize (Op->Asl.Child);
759     }

761     /* If not a variable-length package, check for a zero package length */

763     if ((PackageLengthOp->Asl.ParseOpcode == PARSEOP_INTEGER)      ||
764         (PackageLengthOp->Asl.ParseOpcode == PARSEOP_QWORDCONST)   ||
765         (PackageLengthOp->Asl.ParseOpcode == PARSEOP_ZERO)         ||
766         (PackageLengthOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG))
767     {
768         if (!PackageLength)
769         {
770             /* No length AND no initializer list -- issue a remark */

772             AslError (ASL_REMARK, ASL_MSG_PACKAGE_LENGTH,
773                 PackageLengthOp, NULL);

775             /* But go ahead and put the buffer length of zero into the AML */
776         }
777     }

779     /*
780      * If the PackageLength is a constant <= 255, we can change the
781      * AML opcode from VarPackage to a simple (ACPI 1.0) Package opcode.
782      */
783     if (((Op->Asl.Child->Asl.ParseOpcode == PARSEOP_INTEGER) &&
784             (Op->Asl.Child->Asl.Value.Integer <= 255))  ||
785         (Op->Asl.Child->Asl.ParseOpcode == PARSEOP_ONE) ||
```

```
786         (Op->Asl.Child->Asl.ParseOpcode == PARSEOP_ONES)||
787         (Op->Asl.Child->Asl.ParseOpcode == PARSEOP_ZERO))
788     {
789         Op->Asl.AmlOpcode = AML_PACKAGE_OP;
790         Op->Asl.ParseOpcode = PARSEOP_PACKAGE;

792         /*
793          * Just set the package size node to be the package length, regardless
794          * of whether it was previously an integer or a default_arg placeholder
795          */
796         PackageLengthOp->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
797         PackageLengthOp->Asl.AmlLength = 1;
798         PackageLengthOp->Asl.ParseOpcode = PARSEOP_RAW_DATA;
799         PackageLengthOp->Asl.Value.Integer = PackageLength;
800     }

802     /* Remaining nodes are handled via the tree walk */
803 }


806 /*******************************************************************************
807  *
808  * FUNCTION:    OpnDoLoadTable
809  *
810  * PARAMETERS:  Op        - The parent parse node
811  *
812  * RETURN:      None
813  *
814  * DESCRIPTION: Construct the AML operands for the LOADTABLE ASL keyword.
815  *
816  ******************************************************************************/

818 static void
819 OpnDoLoadTable (
820     ACPI_PARSE_OBJECT       *Op)
821 {
822     ACPI_PARSE_OBJECT       *Next;


825     /* Opcode is parent node */
826     /* First child is the table signature */

828     Next = Op->Asl.Child;

830     /* Second child is the OEM ID*/

832     Next = Next->Asl.Next;

834     /* Third child is the OEM table ID */

836     Next = Next->Asl.Next;

838     /* Fourth child is the RootPath string */

840     Next = Next->Asl.Next;
841     if (Next->Asl.ParseOpcode == PARSEOP_ZERO)
842     {
843         Next->Asl.ParseOpcode    = PARSEOP_STRING_LITERAL;
844         Next->Asl.Value.String   = "\\";
845         Next->Asl.AmlLength      = 2;
846         OpcGenerateAmlOpcode (Next);
847     }

849 #ifdef ASL_FUTURE_IMPLEMENTATION

851     /* TBD: NOT IMPLEMENTED */
```



```
852     /* Fifth child is the [optional] ParameterPathString */
853     /* Sixth child is the [optional] ParameterData */

855     Next = Next->Asl.Next;
856     if (Next->Asl.ParseOpcode == DEFAULT_ARG)
857     {
858         Next->Asl.AmlLength = 1;
859         Next->Asl.ParseOpcode = ZERO;
860         OpcGenerateAmlOpcode (Next);
861     }


864     Next = Next->Asl.Next;
865     if (Next->Asl.ParseOpcode == DEFAULT_ARG)
866     {
867         Next->Asl.AmlLength = 1;
868         Next->Asl.ParseOpcode = ZERO;
869         OpcGenerateAmlOpcode (Next);
870     }
871 #endif
872 }


875 /*******************************************************************************
876  *
877  * FUNCTION:    OpnDoDefinitionBlock
878  *
879  * PARAMETERS:  Op        - The parent parse node
880  *
881  * RETURN:      None
882  *
883  * DESCRIPTION: Construct the AML operands for the DEFINITIONBLOCK ASL keyword
884  *
885  ******************************************************************************/

887 static void
888 OpnDoDefinitionBlock (
889     ACPI_PARSE_OBJECT       *Op)
890 {
891     ACPI_PARSE_OBJECT       *Child;
892     ACPI_SIZE               Length;
893     UINT32                  i;
894     char                    *Filename;


897     /*
898      * These nodes get stuffed into the table header. They are special
899      * cased when the table is written to the output file.
900      *
901      * Mark all of these nodes as non-usable so they won't get output
902      * as AML opcodes!
903      */

905     /* Get AML filename. Use it if non-null */

907     Child = Op->Asl.Child;
908     if (Child->Asl.Value.Buffer  &&
909         *Child->Asl.Value.Buffer &&
910         (Gbl_UseDefaultAmlFilename))
911     {
912         /*
913          * We will use the AML filename that is embedded in the source file
914          * for the output filename.
915          */
916         Filename = ACPI_ALLOCATE (strlen (Gbl_DirectoryPath) +
917                     strlen ((char *) Child->Asl.Value.Buffer) + 1);
```

```
919         /* Prepend the current directory path */
921         strcpy (Filename, Gbl_DirectoryPath);
922         strcat (Filename, (char *) Child->Asl.Value.Buffer);
924         Gbl_OutputFilenamePrefix = Filename;
925         UtConvertBackslashes (Gbl_OutputFilenamePrefix);
926     }
927     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
929     /* Signature */
931     Child = Child->Asl.Next;
932     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
933     if (Child->Asl.Value.String)
934     {
935         Gbl_TableSignature = Child->Asl.Value.String;
936         if (ACPI_STRLEN (Gbl_TableSignature) != 4)
937         {
938             AslError (ASL_ERROR, ASL_MSG_TABLE_SIGNATURE, Child,
939                 "Length not exactly 4");
940         }
942         for (i = 0; i < 4; i++)
943         {
944             if (!isalnum ((int) Gbl_TableSignature[i]))
945             {
946                 AslError (ASL_ERROR, ASL_MSG_TABLE_SIGNATURE, Child,
947                     "Contains non-alphanumeric characters");
948             }
949         }
950     }
952     /* Revision */
954     Child = Child->Asl.Next;
955     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
956     /*
957      * We used the revision to set the integer width earlier
958      */
960     /* OEMID */
962     Child = Child->Asl.Next;
963     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
965     /* OEM TableID */
967     Child = Child->Asl.Next;
968     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
969     if (Child->Asl.Value.String)
970     {
971         Length = ACPI_STRLEN (Child->Asl.Value.String);
972         Gbl_TableId = AcpiOsAllocate (Length + 1);
973         ACPI_STRCPY (Gbl_TableId, Child->Asl.Value.String);
975         /*
976          * Convert anything non-alphanumeric to an underscore. This
977          * allows us to use the TableID to generate unique C symbols.
978          */
979         for (i = 0; i < Length; i++)
980         {
981             if (!isalnum ((int) Gbl_TableId[i]))
982             {
983                 Gbl_TableId[i] = '_';
```



```
984             }
985         }
986     }
988     /* OEM Revision */
990     Child = Child->Asl.Next;
991     Child->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
992 }
995 /*******************************************************************************
996  *
997  * FUNCTION:    UtGetArg
998  *
999  * PARAMETERS:  Op              - Get an argument for this op
1000 *              Argn            - Nth argument to get
1001 *
1002 * RETURN:      The argument (as an Op object). NULL if argument does not exist
1003 *
1004 * DESCRIPTION: Get the specified op's argument (peer)
1005 *
1006 ******************************************************************************/
1008 ACPI_PARSE_OBJECT *
1009 UtGetArg (
1010     ACPI_PARSE_OBJECT       *Op,
1011     UINT32                  Argn)
1012 {
1013     ACPI_PARSE_OBJECT       *Arg = NULL;
1016     /* Get the requested argument object */
1018     Arg = Op->Asl.Child;
1019     while (Arg && Argn)
1020     {
1021         Argn--;
1022         Arg = Arg->Asl.Next;
1023     }
1025     return (Arg);
1026 }
1029 /*******************************************************************************
1030 *
1031 * FUNCTION:    OpnAttachNameToNode
1032 *
1033 * PARAMETERS:  Op        - The parent parse node
1034 *
1035 * RETURN:      None
1036 *
1037 * DESCRIPTION: For the named ASL/AML operators, get the actual name from the
1038 *              argument list and attach it to the parent node so that we
1039 *              can get to it quickly later.
1040 *
1041 ******************************************************************************/
1043 static void
1044 OpnAttachNameToNode (
1045     ACPI_PARSE_OBJECT       *Op)
1046 {
1047     ACPI_PARSE_OBJECT       *Child = NULL;
```

```
    if (Op->Asl.ParseOpcode == PARSEOP_EXTERNAL)
    {
        Child = UtGetArg (Op, 0);
    }
    else switch (Op->Asl.AmlOpcode)
    {
    case AML_DATA_REGION_OP:
    case AML_DEVICE_OP:
    case AML_EVENT_OP:
    case AML_METHOD_OP:
    case AML_MUTEX_OP:
    case AML_REGION_OP:
    case AML_POWER_RES_OP:
    case AML_PROCESSOR_OP:
    case AML_THERMAL_ZONE_OP:
    case AML_NAME_OP:
    case AML_SCOPE_OP:

        Child = UtGetArg (Op, 0);
        break;

    case AML_ALIAS_OP:

        Child = UtGetArg (Op, 1);
        break;

    case AML_CREATE_BIT_FIELD_OP:
    case AML_CREATE_BYTE_FIELD_OP:
    case AML_CREATE_WORD_FIELD_OP:
    case AML_CREATE_DWORD_FIELD_OP:
    case AML_CREATE_QWORD_FIELD_OP:

        Child = UtGetArg (Op, 2);
        break;

    case AML_CREATE_FIELD_OP:

        Child = UtGetArg (Op, 3);
        break;

    case AML_BANK_FIELD_OP:
    case AML_INDEX_FIELD_OP:
    case AML_FIELD_OP:

        return;

    default:

        return;
    }

    if (Child)
    {
        UtAttachNamepathToOwner (Op, Child);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    OpnGenerateAmlOperands
 *
 * PARAMETERS:  Op        - The parent parse node
 *
 * RETURN:      None
 *
```



```
 * DESCRIPTION: Prepare nodes to be output as AML data and operands. The more
 *              complex AML opcodes require processing of the child nodes
 *              (arguments/operands).
 *
 ******************************************************************************/

void
OpnGenerateAmlOperands (
    ACPI_PARSE_OBJECT       *Op)
{


    if (Op->Asl.AmlOpcode == AML_RAW_DATA_BYTE)
    {
        return;
    }

    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_DEFINITIONBLOCK:

        OpnDoDefinitionBlock (Op);
        break;

    case PARSEOP_METHOD:

        OpnDoMethod (Op);
        break;

    case PARSEOP_MUTEX:

        OpnDoMutex (Op);
        break;

    case PARSEOP_FIELD:

        OpnDoField (Op);
        break;

    case PARSEOP_INDEXFIELD:

        OpnDoIndexField (Op);
        break;

    case PARSEOP_BANKFIELD:

        OpnDoBankField (Op);
        break;

    case PARSEOP_BUFFER:

        OpnDoBuffer (Op);
        break;

    case PARSEOP_LOADTABLE:

        OpnDoLoadTable (Op);
        break;

    case PARSEOP_OPERATIONREGION:

        OpnDoRegion (Op);
        break;

    case PARSEOP_RESOURCETEMPLATE:
```

```
1182         RsDoResourceTemplate (Op);
1183         break;

1185     case PARSEOP_NAMESEG:
1186     case PARSEOP_NAMESTRING:
1187     case PARSEOP_METHODCALL:
1188     case PARSEOP_STRING_LITERAL:

1190         break;

1192     default:

1194         break;
1195     }

1197     /* TBD: move */

1199     OpnAttachNameToNode (Op);
1200 }
```

```
**********************************************************
   26783 Thu Dec 26 13:48:31 2013
new/usr/src/common/acpica/compiler/aslopt.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslopt- Compiler optimizations
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#include "acparser.h"
#include "amlcode.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslopt")


static UINT32 OptTotal = 0;

/* Local prototypes */
```



```
static ACPI_STATUS
OptSearchToRoot (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState,
    ACPI_NAMESPACE_NODE     *CurrentNode,
    ACPI_NAMESPACE_NODE     *TargetNode,
    ACPI_BUFFER             *TargetPath,
    char                    **NewPath);

static ACPI_STATUS
OptBuildShortestPath (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState,
    ACPI_NAMESPACE_NODE     *CurrentNode,
    ACPI_NAMESPACE_NODE     *TargetNode,
    ACPI_BUFFER             *CurrentPath,
    ACPI_BUFFER             *TargetPath,
    ACPI_SIZE               AmlNameStringLength,
    UINT8                   IsDeclaration,
    char                    **ReturnNewPath);

static ACPI_STATUS
OptOptimizeNameDeclaration (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState,
    ACPI_NAMESPACE_NODE     *CurrentNode,
    ACPI_NAMESPACE_NODE     *TargetNode,
    char                    *AmlNameString,
    char                    **NewPath);


/*******************************************************************************
 *
 * FUNCTION:    OptSearchToRoot
 *
 * PARAMETERS:  Op                  - Current parser op
 *              WalkState           - Current state
 *              CurrentNode         - Where we are in the namespace
 *              TargetNode          - Node to which we are referring
 *              TargetPath          - External full path to the target node
 *              NewPath             - Where the optimized path is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Attempt to optimize a reference to a single 4-character ACPI
 *              name utilizing the search-to-root name resolution algorithm
 *              that is used by AML interpreters.
 *
 ******************************************************************************/

static ACPI_STATUS
OptSearchToRoot (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState,
    ACPI_NAMESPACE_NODE     *CurrentNode,
    ACPI_NAMESPACE_NODE     *TargetNode,
    ACPI_BUFFER             *TargetPath,
    char                    **NewPath)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_GENERIC_STATE      ScopeInfo;
    ACPI_STATUS             Status;
    char                    *Path;
```

```
126     ACPI_FUNCTION_NAME (OptSearchToRoot);


129     /*
130      * Check if search-to-root can be utilized. Use the last NameSeg of
131      * the NamePath and 1) See if can be found and 2) If found, make
132      * sure that it is the same node that we want. If there is another
133      * name in the search path before the one we want, the nodes will
134      * not match, and we cannot use this optimization.
135      */
136     Path = &(((char *) TargetPath->Pointer)[TargetPath->Length -
137                                             ACPI_NAME_SIZE]),
138     ScopeInfo.Scope.Node = CurrentNode;

140     /* Lookup the NameSeg using SEARCH_PARENT (search-to-root) */

142     Status = AcpiNsLookup (&ScopeInfo, Path, ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
143                     ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPEN_SCOPE,
144                     WalkState, &(Node));
145     if (ACPI_FAILURE (Status))
146     {
147         return (Status);
148     }

150     /*
151      * We found the name, but we must check to make sure that the node
152      * matches. Otherwise, there is another identical name in the search
153      * path that precludes the use of this optimization.
154      */
155     if (Node != TargetNode)
156     {
157         /*
158          * This means that another object with the same name was found first,
159          * and we cannot use this optimization.
160          */
161         return (AE_NOT_FOUND);
162     }

164     /* Found the node, we can use this optimization */

166     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
167         "NAMESEG:   %-24s", Path));

169     /* We must allocate a new string for the name (TargetPath gets deleted) */

171     *NewPath = ACPI_ALLOCATE_ZEROED (ACPI_NAME_SIZE + 1);
172     ACPI_STRCPY (*NewPath, Path);

174     if (ACPI_STRNCMP (*NewPath, "_T_", 3))
175     {
176         AslError (ASL_OPTIMIZATION, ASL_MSG_SINGLE_NAME_OPTIMIZATION, Op,
177                 *NewPath);
178     }

180     return (AE_OK);
181 }


184 /*******************************************************************************
185  *
186  * FUNCTION:    OptBuildShortestPath
187  *
188  * PARAMETERS:  Op                  - Current parser op
189  *              WalkState           - Current state
190  *              CurrentNode         - Where we are in the namespace
191  *              TargetNode          - Node to which we are referring
```



```
192  *              CurrentPath         - External full path to the current node
193  *              TargetPath          - External full path to the target node
194  *              AmlNameStringLength - Length of the original namepath
195  *              IsDeclaration       - TRUE for declaration, FALSE for reference
196  *              ReturnNewPath       - Where the optimized path is returned
197  *
198  * RETURN:      Status
199  *
200  * DESCRIPTION: Build an optimal NamePath using carats
201  *
202  ******************************************************************************/

204 static ACPI_STATUS
205 OptBuildShortestPath (
206     ACPI_PARSE_OBJECT       *Op,
207     ACPI_WALK_STATE         *WalkState,
208     ACPI_NAMESPACE_NODE     *CurrentNode,
209     ACPI_NAMESPACE_NODE     *TargetNode,
210     ACPI_BUFFER             *CurrentPath,
211     ACPI_BUFFER             *TargetPath,
212     ACPI_SIZE               AmlNameStringLength,
213     UINT8                   IsDeclaration,
214     char                    **ReturnNewPath)
215 {
216     UINT32                  NumCommonSegments;
217     UINT32                  MaxCommonSegments;
218     UINT32                  Index;
219     UINT32                  NumCarats;
220     UINT32                  i;
221     char                    *NewPath;
222     char                    *NewPathExternal;
223     ACPI_NAMESPACE_NODE     *Node;
224     ACPI_GENERIC_STATE      ScopeInfo;
225     ACPI_STATUS             Status;
226     BOOLEAN                 SubPath = FALSE;


229     ACPI_FUNCTION_NAME (OptBuildShortestPath);


232     ScopeInfo.Scope.Node = CurrentNode;

234     /*
235      * Determine the maximum number of NameSegs that the Target and Current path
236      * can possibly have in common. (To optimize, we have to have at least 1)
237      *
238      * Note: The external NamePath string lengths are always a multiple of 5
239      * (ACPI_NAME_SIZE + separator)
240      */
241     MaxCommonSegments = TargetPath->Length / ACPI_PATH_SEGMENT_LENGTH;
242     if (CurrentPath->Length < TargetPath->Length)
243     {
244         MaxCommonSegments = CurrentPath->Length / ACPI_PATH_SEGMENT_LENGTH;
245     }

247     /*
248      * Determine how many NameSegs the two paths have in common.
249      * (Starting from the root)
250      */
251     for (NumCommonSegments = 0;
252          NumCommonSegments < MaxCommonSegments;
253          NumCommonSegments++)
254     {
255         /* Compare two single NameSegs */

257         if (!ACPI_COMPARE_NAME (
```

```
258             &((char *) TargetPath->Pointer)[
259                 (NumCommonSegments * ACPI_PATH_SEGMENT_LENGTH) + 1],
260             &((char *) CurrentPath->Pointer)[
261                 (NumCommonSegments * ACPI_PATH_SEGMENT_LENGTH) + 1]))
262         {
263             /* Mismatch */

265             break;
266         }
267     }

269     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " COMMON: %u",
270         NumCommonSegments));

272     /* There must be at least 1 common NameSeg in order to optimize */

274     if (NumCommonSegments == 0)
275     {
276         return (AE_NOT_FOUND);
277     }

279     if (NumCommonSegments == MaxCommonSegments)
280     {
281         if (CurrentPath->Length == TargetPath->Length)
282         {
283             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " SAME PATH"));
284             return (AE_NOT_FOUND);
285         }
286         else
287         {
288             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " SUBPATH"));
289             SubPath = TRUE;
290         }
291     }

293     /* Determine how many prefix Carats are required */

295     NumCarats = (CurrentPath->Length / ACPI_PATH_SEGMENT_LENGTH) -
296                 NumCommonSegments;

298     /*
299      * Construct a new target string
300      */
301     NewPathExternal = ACPI_ALLOCATE_ZEROED (
302         TargetPath->Length + NumCarats + 1);

304     /* Insert the Carats into the Target string */

306     for (i = 0; i < NumCarats; i++)
307     {
308         NewPathExternal[i] = AML_PARENT_PREFIX;
309     }

311     /*
312      * Copy only the necessary (optimal) segments from the original
313      * target string
314      */
315     Index = (NumCommonSegments * ACPI_PATH_SEGMENT_LENGTH) + 1;

317     /* Special handling for exact subpath in a name declaration */

319     if (IsDeclaration && SubPath && (CurrentPath->Length > TargetPath->Length))
320     {
321         /*
322          * The current path is longer than the target, and the target is a
323          * subpath of the current path. We must include one more NameSeg of
```



```
324          * the target path
325          */
326         Index -= ACPI_PATH_SEGMENT_LENGTH;

328         /* Special handling for Scope() operator */

330         if (Op->Asl.AmlOpcode == AML_SCOPE_OP)
331         {
332             NewPathExternal[i] = AML_PARENT_PREFIX;
333             i++;
334             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, "(EXTRA ^)"));
335         }
336     }

338     /* Make sure we haven't gone off the end of the target path */

340     if (Index > TargetPath->Length)
341     {
342         Index = TargetPath->Length;
343     }

345     ACPI_STRCPY (&NewPathExternal[i], &((char *) TargetPath->Pointer)[Index]);
346     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " %-24s", NewPathExternal));

348     /*
349      * Internalize the new target string and check it against the original
350      * string to make sure that this is in fact an optimization. If the
351      * original string is already optimal, there is no point in continuing.
352      */
353     Status = AcpiNsInternalizeName (NewPathExternal, &NewPath);
354     if (ACPI_FAILURE (Status))
355     {
356         AslCoreSubsystemError (Op, Status, "Internalizing new NamePath",
357             ASL_NO_ABORT);
358         ACPI_FREE (NewPathExternal);
359         return (Status);
360     }

362     if (ACPI_STRLEN (NewPath) >= AmlNameStringLength)
363     {
364         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
365             " NOT SHORTER (New %u old %u)",
366             (UINT32) ACPI_STRLEN (NewPath), (UINT32) AmlNameStringLength));
367         ACPI_FREE (NewPathExternal);
368         return (AE_NOT_FOUND);
369     }

371     /*
372      * Check to make sure that the optimization finds the node we are
373      * looking for. This is simply a sanity check on the new
374      * path that has been created.
375      */
376     Status = AcpiNsLookup (&ScopeInfo,  NewPath,
377                     ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
378                     ACPI_NS_DONT_OPEN_SCOPE, WalkState, &(Node));
379     if (ACPI_SUCCESS (Status))
380     {
381         /* Found the namepath, but make sure the node is correct */

383         if (Node == TargetNode)
384         {
385             /* The lookup matched the node, accept this optimization */

387             AslError (ASL_OPTIMIZATION, ASL_MSG_NAME_OPTIMIZATION,
388                 Op, NewPathExternal);
389             *ReturnNewPath = NewPath;
```

```
390         }
391         else
392         {
393             /* Node is not correct, do not use this optimization */

395             Status = AE_NOT_FOUND;
396             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " ***** WRONG NODE"));
397             AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Op,
398                 "Not using optimized name - found wrong node");
399         }
400     }
401     else
402     {
403         /* The lookup failed, we obviously cannot use this optimization */

405         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " ***** NOT FOUND"));
406         AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Op,
407             "Not using optimized name - did not find node");
408     }

410     ACPI_FREE (NewPathExternal);
411     return (Status);
412 }


415 /*******************************************************************************
416  *
417  * FUNCTION:    OptOptimizeNameDeclaration
418  *
419  * PARAMETERS:  Op                  - Current parser op
420  *              WalkState           - Current state
421  *              CurrentNode         - Where we are in the namespace
422  *              AmlNameString       - Unoptimized namepath
423  *              NewPath             - Where the optimized path is returned
424  *
425  * RETURN:      Status. AE_OK If path is optimized
426  *
427  * DESCRIPTION: Perform a simple optimization of removing an extraneous
428  *              backslash prefix if we are already at the root scope.
429  *
430  ******************************************************************************/

432 static ACPI_STATUS
433 OptOptimizeNameDeclaration (
434     ACPI_PARSE_OBJECT       *Op,
435     ACPI_WALK_STATE         *WalkState,
436     ACPI_NAMESPACE_NODE     *CurrentNode,
437     ACPI_NAMESPACE_NODE     *TargetNode,
438     char                    *AmlNameString,
439     char                    **NewPath)
440 {
441     ACPI_STATUS             Status;
442     char                    *NewPathExternal;
443     ACPI_NAMESPACE_NODE     *Node;


446     ACPI_FUNCTION_TRACE (OptOptimizeNameDeclaration);


449     if (((CurrentNode == AcpiGbl_RootNode) ||
450         (Op->Common.Parent->Asl.ParseOpcode == PARSEOP_DEFINITIONBLOCK)) &&
451             (ACPI_IS_ROOT_PREFIX (AmlNameString[0])))
452     {
453         /*
454          * The current scope is the root, and the namepath has a root prefix
455          * that is therefore extraneous. Remove it.
```



```
456          */
457         *NewPath = &AmlNameString[1];

459         /* Debug output */

461         Status = AcpiNsExternalizeName (ACPI_UINT32_MAX, *NewPath,
462                     NULL, &NewPathExternal);
463         if (ACPI_FAILURE (Status))
464         {
465             AslCoreSubsystemError (Op, Status, "Externalizing NamePath",
466                 ASL_NO_ABORT);
467             return (Status);
468         }

470         /*
471          * Check to make sure that the optimization finds the node we are
472          * looking for. This is simply a sanity check on the new
473          * path that has been created.
474          *
475          * We know that we are at the root, so NULL is used for the scope.
476          */
477         Status = AcpiNsLookup (NULL, *NewPath,
478                         ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
479                         ACPI_NS_DONT_OPEN_SCOPE, WalkState, &(Node));
480         if (ACPI_SUCCESS (Status))
481         {
482             /* Found the namepath, but make sure the node is correct */

484             if (Node == TargetNode)
485             {
486                 /* The lookup matched the node, accept this optimization */

488                 AslError (ASL_OPTIMIZATION, ASL_MSG_NAME_OPTIMIZATION,
489                     Op, NewPathExternal);

491                 ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
492                     "AT ROOT:   %-24s", NewPathExternal));
493             }
494             else
495             {
496                 /* Node is not correct, do not use this optimization */

498                 Status = AE_NOT_FOUND;
499                 ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
500                     " ***** WRONG NODE"));
501                 AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Op,
502                     "Not using optimized name - found wrong node");
503             }
504         }
505         else
506         {
507             /* The lookup failed, we obviously cannot use this optimization */

509             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
510                 " ***** NOT FOUND"));
511             AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Op,
512                 "Not using optimized name - did not find node");
513         }

515         ACPI_FREE (NewPathExternal);
516         return (Status);
517     }

519     /* Could not optimize */

521     return (AE_NOT_FOUND);
```

```
522 }


525 /*******************************************************************************
526  *
527  * FUNCTION:    OptOptimizeNamePath
528  *
529  * PARAMETERS:  Op                  - Current parser op
530  *              Flags               - Opcode info flags
531  *              WalkState           - Current state
532  *              AmlNameString       - Unoptimized namepath
533  *              TargetNode          - Node to which AmlNameString refers
534  *
535  * RETURN:      None. If path is optimized, the Op is updated with new path
536  *
537  * DESCRIPTION: Optimize a Named Declaration or Reference to the minimal length.
538  *              Must take into account both the current location in the
539  *              namespace and the actual reference path.
540  *
541  ******************************************************************************/

543 void
544 OptOptimizeNamePath (
545     ACPI_PARSE_OBJECT       *Op,
546     UINT32                  Flags,
547     ACPI_WALK_STATE         *WalkState,
548     char                    *AmlNameString,
549     ACPI_NAMESPACE_NODE     *TargetNode)
550 {
551     ACPI_STATUS             Status;
552     ACPI_BUFFER             TargetPath;
553     ACPI_BUFFER             CurrentPath;
554     ACPI_SIZE               AmlNameStringLength;
555     ACPI_NAMESPACE_NODE     *CurrentNode;
556     char                    *ExternalNameString;
557     char                    *NewPath = NULL;
558     ACPI_SIZE               HowMuchShorter;
559     ACPI_PARSE_OBJECT       *NextOp;


562     ACPI_FUNCTION_TRACE (OptOptimizeNamePath);


565     /* This is an optional optimization */

567     if (!Gbl_ReferenceOptimizationFlag)
568     {
569         return_VOID;
570     }

572     /* Various required items */

574     if (!TargetNode || !WalkState || !AmlNameString || !Op->Common.Parent)
575     {
576         return_VOID;
577     }

579     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
580         "PATH OPTIMIZE: Line %5d ParentOp [%12.12s] ThisOp [%12.12s] ",
581         Op->Asl.LogicalLineNumber,
582         AcpiPsGetOpcodeName (Op->Common.Parent->Common.AmlOpcode),
583         AcpiPsGetOpcodeName (Op->Common.AmlOpcode)));

585     if (!(Flags & (AML_NAMED | AML_CREATE)))
586     {
587         if (Op->Asl.CompileFlags & NODE_IS_NAME_DECLARATION)
```



```
588         {
589             /* We don't want to fuss with actual name declaration nodes here */

591             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
592                 "******* NAME DECLARATION\n"));
593             return_VOID;
594         }
595     }

597     /*
598      * The original path must be longer than one NameSeg (4 chars) for there
599      * to be any possibility that it can be optimized to a shorter string
600      */
601     AmlNameStringLength = ACPI_STRLEN (AmlNameString);
602     if (AmlNameStringLength <= ACPI_NAME_SIZE)
603     {
604         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
605             "NAMESEG %4.4s\n", AmlNameString));
606         return_VOID;
607     }

609     /*
610      * We need to obtain the node that represents the current scope -- where
611      * we are right now in the namespace. We will compare this path
612      * against the Namepath, looking for commonality.
613      */
614     CurrentNode = AcpiGbl_RootNode;
615     if (WalkState->ScopeInfo)
616     {
617         CurrentNode = WalkState->ScopeInfo->Scope.Node;
618     }

620     if (Flags & (AML_NAMED | AML_CREATE))
621     {
622         /* This is the declaration of a new name */

624         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, "NAME\n"));

626         /*
627          * The node of interest is the parent of this node (the containing
628          * scope). The actual namespace node may be up more than one level
629          * of parse op or it may not exist at all (if we traverse back
630          * up to the root.)
631          */
632         NextOp = Op->Asl.Parent;
633         while (NextOp && (!NextOp->Asl.Node))
634         {
635             NextOp = NextOp->Asl.Parent;
636         }
637         if (NextOp && NextOp->Asl.Node)
638         {
639             CurrentNode = NextOp->Asl.Node;
640         }
641         else
642         {
643             CurrentNode = AcpiGbl_RootNode;
644         }
645     }
646     else
647     {
648         /* This is a reference to an existing named object */

650         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, "REFERENCE\n"));
651     }

653     /*
```

```
654      * Obtain the full paths to the two nodes that we are interested in
655      * (Target and current namespace location) in external
656      * format -- something we can easily manipulate
657      */
658     TargetPath.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
659     Status = AcpiNsHandleToPathname (TargetNode, &TargetPath);
660     if (ACPI_FAILURE (Status))
661     {
662         AslCoreSubsystemError (Op, Status, "Getting Target NamePath",
663             ASL_NO_ABORT);
664         return_VOID;
665     }
666     TargetPath.Length--;    /* Subtract one for null terminator */

668     /* CurrentPath is the path to this scope (where we are in the namespace) */

670     CurrentPath.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
671     Status = AcpiNsHandleToPathname (CurrentNode, &CurrentPath);
672     if (ACPI_FAILURE (Status))
673     {
674         AslCoreSubsystemError (Op, Status, "Getting Current NamePath",
675             ASL_NO_ABORT);
676         return_VOID;
677     }
678     CurrentPath.Length--;   /* Subtract one for null terminator */

680     /* Debug output only */

682     Status = AcpiNsExternalizeName (ACPI_UINT32_MAX, AmlNameString,
683                 NULL, &ExternalNameString);
684     if (ACPI_FAILURE (Status))
685     {
686         AslCoreSubsystemError (Op, Status, "Externalizing NamePath",
687             ASL_NO_ABORT);
688         return_VOID;
689     }

691     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
692         "CURRENT SCOPE: (%2u) %-37s FULL PATH TO NAME: (%2u) %-32s ACTUAL AML:%-
693         (UINT32) CurrentPath.Length, (char *) CurrentPath.Pointer,
694         (UINT32) TargetPath.Length, (char *) TargetPath.Pointer,
695         ExternalNameString));

697     ACPI_FREE (ExternalNameString);

699     /*
700      * Attempt an optmization depending on the type of namepath
701      */
702     if (Flags & (AML_NAMED | AML_CREATE))
703     {
704         /*
705          * This is a named opcode and the namepath is a name declaration, not
706          * a reference.
707          */
708         Status = OptOptimizeNameDeclaration (Op, WalkState, CurrentNode,
709                     TargetNode, AmlNameString, &NewPath);
710         if (ACPI_FAILURE (Status))
711         {
712             /*
713              * 2) now attempt to
714              *    optimize the namestring with carats (up-arrow)
715              */
716             Status = OptBuildShortestPath (Op, WalkState, CurrentNode,
717                             TargetNode, &CurrentPath, &TargetPath,
718                             AmlNameStringLength, 1, &NewPath);
719         }
```



```
720     }
721     else
722     {
723         /*
724          * This is a reference to an existing named object
725          *
726          * 1) Check if search-to-root can be utilized using the last
727          *    NameSeg of the NamePath
728          */
729         Status = OptSearchToRoot (Op, WalkState, CurrentNode,
730                         TargetNode, &TargetPath, &NewPath);
731         if (ACPI_FAILURE (Status))
732         {
733             /*
734              * 2) Search-to-root could not be used, now attempt to
735              *    optimize the namestring with carats (up-arrow)
736              */
737             Status = OptBuildShortestPath (Op, WalkState, CurrentNode,
738                             TargetNode, &CurrentPath, &TargetPath,
739                             AmlNameStringLength, 0, &NewPath);
740         }
741     }

743     /*
744      * Success from above indicates that the NamePath was successfully
745      * optimized. We need to update the parse op with the new name
746      */
747     if (ACPI_SUCCESS (Status))
748     {
749         HowMuchShorter = (AmlNameStringLength - ACPI_STRLEN (NewPath));
750         OptTotal += HowMuchShorter;

752         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
753             " REDUCED BY %2u (TOTAL SAVED %2u)",
754             (UINT32) HowMuchShorter, OptTotal));

756         if (Flags & AML_NAMED)
757         {
758             if (Op->Asl.AmlOpcode == AML_ALIAS_OP)
759             {
760                 /*
761                  * ALIAS is the only oddball opcode, the name declaration
762                  * (alias name) is the second operand
763                  */
764                 Op->Asl.Child->Asl.Next->Asl.Value.String = NewPath;
765                 Op->Asl.Child->Asl.Next->Asl.AmlLength = ACPI_STRLEN (NewPath);
766             }
767             else
768             {
769                 Op->Asl.Child->Asl.Value.String = NewPath;
770                 Op->Asl.Child->Asl.AmlLength = ACPI_STRLEN (NewPath);
771             }
772         }
773         else if (Flags & AML_CREATE)
774         {
775             /* Name must appear as the last parameter */

777             NextOp = Op->Asl.Child;
778             while (!(NextOp->Asl.CompileFlags & NODE_IS_NAME_DECLARATION))
779             {
780                 NextOp = NextOp->Asl.Next;
781             }
782             /* Update the parse node with the new NamePath */

784             NextOp->Asl.Value.String = NewPath;
785             NextOp->Asl.AmlLength = ACPI_STRLEN (NewPath);
```

```
786         }
787         else
788         {
789             /* Update the parse node with the new NamePath */
790
791             Op->Asl.Value.String = NewPath;
792             Op->Asl.AmlLength = ACPI_STRLEN (NewPath);
793         }
794     }
795     else
796     {
797         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " ALREADY OPTIMAL"));
798     }
799
800     /* Cleanup path buffers */
801
802     ACPI_FREE (TargetPath.Pointer);
803     ACPI_FREE (CurrentPath.Pointer);
804
805     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, "\n"));
806     return_VOID;
807 }
```

```
**********************************************************
   19027 Thu Dec 26 13:48:32 2013
new/usr/src/common/acpica/compiler/asloptions.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: asloptions - compiler command line processing
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #include "aslcompiler.h"
  45 #include "acapps.h"
  46 #include "acdisasm.h"

  48 #define _COMPONENT          ACPI_COMPILER
  49         ACPI_MODULE_NAME    ("asloption")


  52 /* Local prototypes */

  54 static int
  55 AslDoOptions (
  56     int                     argc,
  57     char                    **argv,
  58     BOOLEAN                 IsResponseFile);
```



```
  60 static void
  61 AslMergeOptionTokens (
  62     char                    *InBuffer,
  63     char                    *OutBuffer);

  65 static int
  66 AslDoResponseFile (
  67     char                    *Filename);


  70 #define ASL_TOKEN_SEPARATORS    " \t\n"
  71 #define ASL_SUPPORTED_OPTIONS   "@:b|c|d^D:e:f^gh^i|I:l^m:no|p:P^r:s|t|T+G^v^w|x


  74 /******************************************************************************
  75  *
  76  * FUNCTION:    AslCommandLine
  77  *
  78  * PARAMETERS:  argc/argv
  79  *
  80  * RETURN:      Last argv index
  81  *
  82  * DESCRIPTION: Command line processing
  83  *
  84  *****************************************************************************/

  86 int
  87 AslCommandLine (
  88     int                     argc,
  89     char                    **argv)
  90 {
  91     int                     BadCommandLine = 0;
  92     ACPI_STATUS             Status;


  95     /* Minimum command line contains at least the command and an input file */

  97     if (argc < 2)
  98     {
  99         printf (ACPI_COMMON_SIGNON (ASL_COMPILER_NAME));
 100         Usage ();
 101         exit (1);
 102     }

 104     /* Process all command line options */

 106     BadCommandLine = AslDoOptions (argc, argv, FALSE);

 108     if (Gbl_DoTemplates)
 109     {
 110         Status = DtCreateTemplates (Gbl_TemplateSignature);
 111         if (ACPI_FAILURE (Status))
 112         {
 113             exit (-1);
 114         }
 115         exit (1);
 116     }

 118     /* Next parameter must be the input filename */

 120     if (!argv[AcpiGbl_Optind] &&
 121         !Gbl_DisasmFlag &&
 122         !Gbl_GetAllTables)
 123     {
 124         printf ("Missing input filename\n");
 125         BadCommandLine = TRUE;
```

```
126     }

128     if (Gbl_DoSignon)
129     {
130         printf (ACPI_COMMON_SIGNON (ASL_COMPILER_NAME));
131         if (Gbl_IgnoreErrors)
132         {
133             printf ("Ignoring all errors, forcing AML file generation\n\n");
134         }
135     }

137     if (BadCommandLine)
138     {
139         printf ("Use -h option for help information\n");
140         exit (1);
141     }

143     return (AcpiGbl_Optind);
144 }


147 /******************************************************************************
148  *
149  * FUNCTION:    AslDoOptions
150  *
151  * PARAMETERS:  argc/argv           - Standard argc/argv
152  *              IsResponseFile      - TRUE if executing a response file.
153  *
154  * RETURN:      Status
155  *
156  * DESCRIPTION: Command line option processing
157  *
158  *****************************************************************************/

160 static int
161 AslDoOptions (
162     int                     argc,
163     char                    **argv,
164     BOOLEAN                 IsResponseFile)
165 {
166     ACPI_STATUS             Status;
167     UINT32                  j;


170     /* Get the command line options */

172     while ((j = AcpiGetopt (argc, argv, ASL_SUPPORTED_OPTIONS)) != EOF) switch (
173     {
174     case '@':   /* Begin a response file */

176         if (IsResponseFile)
177         {
178             printf ("Nested command files are not supported\n");
179             return (-1);
180         }

182         if (AslDoResponseFile (AcpiGbl_Optarg))
183         {
184             return (-1);
185         }
186         break;

188     case 'b':   /* Debug output options */

190         switch (AcpiGbl_Optarg[0])
191         {
```



```
192         case 'f':

194             AslCompilerdebug = 1; /* same as yydebug */
195             DtParserdebug = 1;
196             PrParserdebug = 1;
197             break;

199         case 't':

201             break;

203         default:

205             printf ("Unknown option: -b%s\n", AcpiGbl_Optarg);
206             return (-1);
207         }

209         /* Produce debug output file */

211         Gbl_DebugFlag = TRUE;
212         break;

214     case 'c':

216         switch (AcpiGbl_Optarg[0])
217         {
218         case 'r':

220             Gbl_NoResourceChecking = TRUE;
221             break;

223         default:

225             printf ("Unknown option: -c%s\n", AcpiGbl_Optarg);
226             return (-1);
227         }
228         break;

230     case 'd':   /* Disassembler */

232         switch (AcpiGbl_Optarg[0])
233         {
234         case '^':

236             Gbl_DoCompile = FALSE;
237             break;

239         case 'a':

241             Gbl_DoCompile = FALSE;
242             Gbl_DisassembleAll = TRUE;
243             break;

245         case 'b':   /* Do not convert buffers to resource descriptors */

247             AcpiGbl_NoResourceDisassembly = TRUE;
248             break;

250         case 'c':

252             break;

254         default:

256             printf ("Unknown option: -d%s\n", AcpiGbl_Optarg);
257             return (-1);
```

```
258         }
260         Gbl_DisasmFlag = TRUE;
261         break;
263     case 'D':   /* Define a symbol */
265         PrAddDefine (AcpiGbl_Optarg, NULL, TRUE);
266         break;
268     case 'e':   /* External files for disassembler */
270         /* Get entire list of external files */
272         AcpiGbl_Optind--;
274         while (argv[AcpiGbl_Optind] &&
275               (argv[AcpiGbl_Optind][0] != '-'))
276         {
277             Status = AcpiDmAddToExternalFileList (argv[AcpiGbl_Optind]);
278             if (ACPI_FAILURE (Status))
279             {
280                 printf ("Could not add %s to external list\n", argv[AcpiGbl_Opti
281                 return (-1);
282             }
284             AcpiGbl_Optind++;
285         }
286         break;
288     case 'f':
290         switch (AcpiGbl_Optarg[0])
291         {
292         case '^':   /* Ignore errors and force creation of aml file */
294             Gbl_IgnoreErrors = TRUE;
295             break;
297         case 'e':   /* Disassembler: Get external declaration file */
299             if (AcpiGetoptArgument (argc, argv))
300             {
301                 return (-1);
302             }
304             Gbl_ExternalRefFilename = AcpiGbl_Optarg;
305             break;
307         default:
309             printf ("Unknown option: -f%s\n", AcpiGbl_Optarg);
310             return (-1);
311         }
312         break;
314     case 'G':
316         Gbl_CompileGeneric = TRUE;
317         break;
319     case 'g':   /* Get all ACPI tables */
321         Gbl_GetAllTables = TRUE;
322         Gbl_DoCompile = FALSE;
323         break;
```



```
325     case 'h':
327         switch (AcpiGbl_Optarg[0])
328         {
329         case '^':
331             Usage ();
332             exit (0);
334         case 'c':
336             UtDisplayConstantOpcodes ();
337             exit (0);
339         case 'f':
341             AslFilenameHelp ();
342             exit (0);
344         case 'r':
346             /* reserved names */
348             ApDisplayReservedNames ();
349             exit (0);
351         case 't':
353             UtDisplaySupportedTables ();
354             exit (0);
356         default:
358             printf ("Unknown option: -h%s\n", AcpiGbl_Optarg);
359             return (-1);
360         }
362     case 'I':   /* Add an include file search directory */
364         FlAddIncludeDirectory (AcpiGbl_Optarg);
365         break;
367     case 'i':   /* Output AML as an include file */
369         switch (AcpiGbl_Optarg[0])
370         {
371         case 'a':
373             /* Produce assembly code include file */
375             Gbl_AsmIncludeOutputFlag = TRUE;
376             break;
378         case 'c':
380             /* Produce C include file */
382             Gbl_C_IncludeOutputFlag = TRUE;
383             break;
385         case 'n':
387             /* Compiler/Disassembler: Ignore the NOOP operator */
389             AcpiGbl_IgnoreNoopOperator = TRUE;
```

```
390             break;

392         default:

394             printf ("Unknown option: -i%s\n", AcpiGbl_Optarg);
395             return (-1);
396         }
397         break;

399     case 'l':   /* Listing files */

401         switch (AcpiGbl_Optarg[0])
402         {
403         case '^':

405             /* Produce listing file (Mixed source/aml) */

407             Gbl_ListingFlag = TRUE;
408             break;

410         case 'i':

412             /* Produce preprocessor output file */

414             Gbl_PreprocessorOutputFlag = TRUE;
415             break;

417         case 'n':

419             /* Produce namespace file */

421             Gbl_NsOutputFlag = TRUE;
422             break;

424         case 's':

426             /* Produce combined source file */

428             Gbl_SourceOutputFlag = TRUE;
429             break;

431         default:

433             printf ("Unknown option: -l%s\n", AcpiGbl_Optarg);
434             return (-1);
435         }
436         break;

438     case 'm':   /* Set line buffer size */

440         Gbl_LineBufferSize = (UINT32) strtoul (AcpiGbl_Optarg, NULL, 0) * 1024;
441         if (Gbl_LineBufferSize < ASL_DEFAULT_LINE_BUFFER_SIZE)
442         {
443             Gbl_LineBufferSize = ASL_DEFAULT_LINE_BUFFER_SIZE;
444         }
445         printf ("Line Buffer Size: %u\n", Gbl_LineBufferSize);
446         break;

448     case 'n':   /* Parse only */

450         Gbl_ParseOnlyFlag = TRUE;
451         break;

453     case 'o':   /* Control compiler AML optimizations */

455         switch (AcpiGbl_Optarg[0])
```



```
456         {
457         case 'a':

459             /* Disable all optimizations */

461             Gbl_FoldConstants = FALSE;
462             Gbl_IntegerOptimizationFlag = FALSE;
463             Gbl_ReferenceOptimizationFlag = FALSE;
464             break;

466         case 'f':

468             /* Disable folding on "normal" expressions */

470             Gbl_FoldConstants = FALSE;
471             break;

473         case 'i':

475             /* Disable integer optimization to constants */

477             Gbl_IntegerOptimizationFlag = FALSE;
478             break;

480         case 'n':

482             /* Disable named reference optimization */

484             Gbl_ReferenceOptimizationFlag = FALSE;
485             break;

487         case 't':

489             /* Display compile time(s) */

491             Gbl_CompileTimesFlag = TRUE;
492             break;

494         default:

496             printf ("Unknown option: -c%s\n", AcpiGbl_Optarg);
497             return (-1);
498         }
499         break;

501     case 'P':   /* Preprocessor options */

503         switch (AcpiGbl_Optarg[0])
504         {
505         case '^':   /* Proprocess only, emit (.i) file */

507             Gbl_PreprocessOnly = TRUE;
508             Gbl_PreprocessorOutputFlag = TRUE;
509             break;

511         case 'n':   /* Disable preprocessor */

513             Gbl_PreprocessFlag = FALSE;
514             break;

516         default:

518             printf ("Unknown option: -P%s\n", AcpiGbl_Optarg);
519             return (-1);
520         }
521         break;
```

```
523     case 'p':   /* Override default AML output filename */
525         Gbl_OutputFilenamePrefix = AcpiGbl_Optarg;
526         UtConvertBackslashes (Gbl_OutputFilenamePrefix);
528         Gbl_UseDefaultAmlFilename = FALSE;
529         break;
531     case 'r':   /* Override revision found in table header */
533         Gbl_RevisionOverride = (UINT8) strtoul (AcpiGbl_Optarg, NULL, 0);
534         break;
536     case 's':   /* Create AML in a source code file */
538         switch (AcpiGbl_Optarg[0])
539         {
540         case 'a':
542             /* Produce assembly code output file */
544             Gbl_AsmOutputFlag = TRUE;
545             break;
547         case 'c':
549             /* Produce C hex output file */
551             Gbl_C_OutputFlag = TRUE;
552             break;
554         case 'o':
556             /* Produce AML offset table in C */
558             Gbl_C_OffsetTableFlag = TRUE;
559             break;
561         default:
563             printf ("Unknown option: -s%s\n", AcpiGbl_Optarg);
564             return (-1);
565         }
566         break;
568     case 't':   /* Produce hex table output file */
570         switch (AcpiGbl_Optarg[0])
571         {
572         case 'a':
574             Gbl_HexOutputFlag = HEX_OUTPUT_ASM;
575             break;
577         case 'c':
579             Gbl_HexOutputFlag = HEX_OUTPUT_C;
580             break;
582         case 's':
584             Gbl_HexOutputFlag = HEX_OUTPUT_ASL;
585             break;
587         default:
```



```
589             printf ("Unknown option: -t%s\n", AcpiGbl_Optarg);
590             return (-1);
591         }
592         break;
594     case 'T':   /* Create a ACPI table template file */
596         Gbl_DoTemplates = TRUE;
597         Gbl_TemplateSignature = AcpiGbl_Optarg;
598         break;
600     case 'v':   /* Version and verbosity settings */
602         switch (AcpiGbl_Optarg[0])
603         {
604         case '^':
606             printf (ACPI_COMMON_SIGNON (ASL_COMPILER_NAME));
607             exit (0);
609         case 'a':
611             /* Disable all error/warning/remark messages */
613             Gbl_NoErrors = TRUE;
614             break;
616         case 'e':
618             /* Disable all warning/remark messages (errors only) */
620             Gbl_DisplayRemarks = FALSE;
621             Gbl_DisplayWarnings = FALSE;
622             break;
624         case 'i':
625             /*
626              * Support for integrated development environment(s).
627              *
628              * 1) No compiler signon
629              * 2) Send stderr messages to stdout
630              * 3) Less verbose error messages (single line only for each)
631              * 4) Error/warning messages are formatted appropriately to
632              *    be recognized by MS Visual Studio
633              */
634             Gbl_VerboseErrors = FALSE;
635             Gbl_DoSignon = FALSE;
636             Gbl_Files[ASL_FILE_STDERR].Handle = stdout;
637             break;
639         case 'o':
641             Gbl_DisplayOptimizations = TRUE;
642             break;
644         case 'r':
646             Gbl_DisplayRemarks = FALSE;
647             break;
649         case 's':
651             Gbl_DoSignon = FALSE;
652             break;
```

```
654         case 't':

656             Gbl_VerboseTemplates = TRUE;
657             break;

659         case 'w':

661             /* Get the required argument */

663             if (AcpiGetoptArgument (argc, argv))
664             {
665                 return (-1);
666             }

668             Status = AslDisableException (AcpiGbl_Optarg);
669             if (ACPI_FAILURE (Status))
670             {
671                 return (-1);
672             }
673             break;

675         default:

677             printf ("Unknown option: -v%s\n", AcpiGbl_Optarg);
678             return (-1);
679         }
680         break;

682     case 'w': /* Set warning levels */

684         switch (AcpiGbl_Optarg[0])
685         {
686         case '1':

688             Gbl_WarningLevel = ASL_WARNING;
689             break;

691         case '2':

693             Gbl_WarningLevel = ASL_WARNING2;
694             break;

696         case '3':

698             Gbl_WarningLevel = ASL_WARNING3;
699             break;

701         case 'e':

703             Gbl_WarningsAsErrors = TRUE;
704             break;

706         default:

708             printf ("Unknown option: -w%s\n", AcpiGbl_Optarg);
709             return (-1);
710         }
711         break;

713     case 'x':   /* Set debug print output level */

715         AcpiDbgLevel = strtoul (AcpiGbl_Optarg, NULL, 16);
716         break;

718     case 'z':
```



```
720         Gbl_UseOriginalCompilerId = TRUE;
721         break;

723     default:

725         return (-1);
726     }

728     return (0);
729 }


732 /******************************************************************************
733  *
734  * FUNCTION:    AslMergeOptionTokens
735  *
736  * PARAMETERS:  InBuffer            - Input containing an option string
737  *              OutBuffer           - Merged output buffer
738  *
739  * RETURN:      None
740  *
741  * DESCRIPTION: Remove all whitespace from an option string.
742  *
743  *****************************************************************************/

745 static void
746 AslMergeOptionTokens (
747     char                    *InBuffer,
748     char                    *OutBuffer)
749 {
750     char                    *Token;


753     *OutBuffer = 0;

755     Token = strtok (InBuffer, ASL_TOKEN_SEPARATORS);
756     while (Token)
757     {
758         strcat (OutBuffer, Token);
759         Token = strtok (NULL, ASL_TOKEN_SEPARATORS);
760     }
761 }


764 /******************************************************************************
765  *
766  * FUNCTION:    AslDoResponseFile
767  *
768  * PARAMETERS:  Filename        - Name of the response file
769  *
770  * RETURN:      Status
771  *
772  * DESCRIPTION: Open a response file and process all options within.
773  *
774  *****************************************************************************/

776 static int
777 AslDoResponseFile (
778     char                    *Filename)
779 {
780     char                    *argv = StringBuffer2;
781     FILE                    *ResponseFile;
782     int                     OptStatus = 0;
783     int                     Opterr;
784     int                     Optind;
```

```
787     ResponseFile = fopen (Filename, "r");
788     if (!ResponseFile)
789     {
790         printf ("Could not open command file %s, %s\n",
791             Filename, strerror (errno));
792         return (-1);
793     }

795     /* Must save the current GetOpt globals */

797     Opterr = AcpiGbl_Opterr;
798     Optind = AcpiGbl_Optind;

800     /*
801      * Process all lines in the response file. There must be one complete
802      * option per line
803      */
804     while (fgets (StringBuffer, ASL_MSG_BUFFER_SIZE, ResponseFile))
805     {
806         /* Compress all tokens, allowing us to use a single argv entry */

808         AslMergeOptionTokens (StringBuffer, StringBuffer2);

810         /* Process the option */

812         AcpiGbl_Opterr = 0;
813         AcpiGbl_Optind = 0;

815         OptStatus = AslDoOptions (1, &argv, TRUE);
816         if (OptStatus)
817         {
818             printf ("Invalid option in command file %s: %s\n",
819                 Filename, StringBuffer);
820             break;
821         }
822     }

824     /* Restore the GetOpt globals */

826     AcpiGbl_Opterr = Opterr;
827     AcpiGbl_Optind = Optind;

829     fclose (ResponseFile);
830     return (OptStatus);
831 }
```

```
**********************************************************
   23978 Thu Dec 26 13:48:32 2013
new/usr/src/common/acpica/compiler/aslpredef.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslpredef - support for ACPI predefined names
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define ACPI_CREATE_PREDEFINED_TABLE
#define ACPI_CREATE_RESOURCE_TABLE

#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acpredef.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslpredef")


/* Local prototypes */

static void
ApCheckForUnexpectedReturnValue (
```



```
    ACPI_PARSE_OBJECT       *Op,
    ASL_METHOD_INFO         *MethodInfo);

static UINT32
ApCheckForSpecialName (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name);


/******************************************************************************
 *
 * FUNCTION:    ApCheckForPredefinedMethod
 *
 * PARAMETERS:  Op              - A parse node of type "METHOD".
 *              MethodInfo      - Saved info about this method
 *
 * RETURN:      None
 *
 * DESCRIPTION: If method is a predefined name, check that the number of
 *              arguments and the return type (returns a value or not)
 *              is correct.
 *
 *****************************************************************************/

BOOLEAN
ApCheckForPredefinedMethod (
    ACPI_PARSE_OBJECT       *Op,
    ASL_METHOD_INFO         *MethodInfo)
{
    UINT32                      Index;
    UINT32                      RequiredArgCount;
    const ACPI_PREDEFINED_INFO  *ThisName;


    /* Check for a match against the predefined name list */

    Index = ApCheckForPredefinedName (Op, Op->Asl.NameSeg);

    switch (Index)
    {
    case ACPI_NOT_RESERVED_NAME:        /* No underscore or _Txx or _xxx name no
    case ACPI_PREDEFINED_NAME:          /* Resource Name or reserved scope name
    case ACPI_COMPILER_RESERVED_NAME:   /* A _Txx that was not emitted by compil

        /* Just return, nothing to do */
        return (FALSE);


    case ACPI_EVENT_RESERVED_NAME:      /* _Lxx/_Exx/_Wxx/_Qxx methods */

        Gbl_ReservedMethods++;

        /* NumArguments must be zero for all _Lxx/_Exx/_Wxx/_Qxx methods */

        if (MethodInfo->NumArguments != 0)
        {
            sprintf (MsgBuffer, "%s requires %u", Op->Asl.ExternalName, 0);

            AslError (ASL_WARNING, ASL_MSG_RESERVED_ARG_COUNT_HI, Op,
                MsgBuffer);
        }
        break;


    default:
        /*
```

```
127          * Matched a predefined method name - validate the ASL-defined
128          * argument count against the ACPI specification.
129          *
130          * Some methods are allowed to have a "minimum" number of args
131          * (_SCP) because their definition in ACPI has changed over time.
132          */
133         Gbl_ReservedMethods++;
134         ThisName = &AcpiGbl_PredefinedMethods[Index];
135         RequiredArgCount = METHOD_GET_ARG_COUNT (ThisName->Info.ArgumentList);

137         if (MethodInfo->NumArguments != RequiredArgCount)
138         {
139             sprintf (MsgBuffer, "%4.4s requires %u",
140                 ThisName->Info.Name, RequiredArgCount);

142             if (MethodInfo->NumArguments < RequiredArgCount)
143             {
144                 AslError (ASL_WARNING, ASL_MSG_RESERVED_ARG_COUNT_LO, Op,
145                     MsgBuffer);
146             }
147             else if ((MethodInfo->NumArguments > RequiredArgCount) &&
148                 !(ThisName->Info.ArgumentList & ARG_COUNT_IS_MINIMUM))
149             {
150                 AslError (ASL_WARNING, ASL_MSG_RESERVED_ARG_COUNT_HI, Op,
151                     MsgBuffer);
152             }
153         }

155         /*
156          * Check if method returns no value, but the predefined name is
157          * required to return a value
158          */
159         if (MethodInfo->NumReturnNoValue &&
160             ThisName->Info.ExpectedBtypes)
161         {
162             AcpiUtGetExpectedReturnTypes (StringBuffer,
163                 ThisName->Info.ExpectedBtypes);

165             sprintf (MsgBuffer, "%s required for %4.4s",
166                 StringBuffer, ThisName->Info.Name);

168             AslError (ASL_WARNING, ASL_MSG_RESERVED_RETURN_VALUE, Op,
169                 MsgBuffer);
170         }
171         break;
172     }

174     return (TRUE);
175 }


178 /*******************************************************************************
179  *
180  * FUNCTION:    ApCheckForUnexpectedReturnValue
181  *
182  * PARAMETERS:  Op              - A parse node of type "RETURN".
183  *              MethodInfo      - Saved info about this method
184  *
185  * RETURN:      None
186  *
187  * DESCRIPTION: Check for an unexpected return value from a predefined method.
188  *              Invoked for predefined methods that are defined to not return
189  *              any value. If there is a return value, issue a remark, since
190  *              the ASL writer may be confused as to the method definition
191  *              and/or functionality.
192  *
```



```
193  * Note: We ignore all return values of "Zero", since this is what a standalone
194  *       Return() statement will always generate -- so we ignore it here --
195  *       i.e., there is no difference between Return() and Return(Zero).
196  *       Also, a null Return() will be disassembled to return(Zero) -- so, we
197  *       don't want to generate extraneous remarks/warnings for a disassembled
198  *       ASL file.
199  *
200  ******************************************************************************/

202 static void
203 ApCheckForUnexpectedReturnValue (
204     ACPI_PARSE_OBJECT       *Op,
205     ASL_METHOD_INFO         *MethodInfo)
206 {
207     ACPI_PARSE_OBJECT       *ReturnValueOp;


210     /* Ignore Return() and Return(Zero) (they are the same) */

212     ReturnValueOp = Op->Asl.Child;
213     if (ReturnValueOp->Asl.ParseOpcode == PARSEOP_ZERO)
214     {
215         return;
216     }

218     /* We have a valid return value, but the reserved name did not expect it */

220     AslError (ASL_WARNING, ASL_MSG_RESERVED_NO_RETURN_VAL,
221         Op, MethodInfo->Op->Asl.ExternalName);
222 }


225 /*******************************************************************************
226  *
227  * FUNCTION:    ApCheckPredefinedReturnValue
228  *
229  * PARAMETERS:  Op              - A parse node of type "RETURN".
230  *              MethodInfo      - Saved info about this method
231  *
232  * RETURN:      None
233  *
234  * DESCRIPTION: If method is a predefined name, attempt to validate the return
235  *              value. Only "static" types can be validated - a simple return
236  *              of an integer/string/buffer/package or a named reference to
237  *              a static object. Values such as a Localx or Argx or a control
238  *              method invocation are not checked. Issue a warning if there is
239  *              a valid return value, but the reserved method defines no
240  *              return value.
241  *
242  ******************************************************************************/

244 void
245 ApCheckPredefinedReturnValue (
246     ACPI_PARSE_OBJECT       *Op,
247     ASL_METHOD_INFO         *MethodInfo)
248 {
249     UINT32                      Index;
250     ACPI_PARSE_OBJECT           *ReturnValueOp;
251     const ACPI_PREDEFINED_INFO  *ThisName;


254     /* Check parent method for a match against the predefined name list */

256     Index = ApCheckForPredefinedName (MethodInfo->Op,
257                 MethodInfo->Op->Asl.NameSeg);
```

```
259     switch (Index)
260     {
261     case ACPI_EVENT_RESERVED_NAME:      /* _Lxx/_Exx/_Wxx/_Qxx methods */

263         /* No return value expected, warn if there is one */

265         ApCheckForUnexpectedReturnValue (Op, MethodInfo);
266         return;

268     case ACPI_NOT_RESERVED_NAME:        /* No underscore or _Txx or _xxx name no
269     case ACPI_PREDEFINED_NAME:          /* Resource Name or reserved scope name
270     case ACPI_COMPILER_RESERVED_NAME:   /* A _Txx that was not emitted by compil

272         /* Just return, nothing to do */
273         return;

275     default: /* A standard predefined ACPI name */

277         ThisName = &AcpiGbl_PredefinedMethods[Index];
278         if (!ThisName->Info.ExpectedBtypes)
279         {
280             /* No return value expected, warn if there is one */

282             ApCheckForUnexpectedReturnValue (Op, MethodInfo);
283             return;
284         }

286         /* Get the object returned, it is the next argument */

288         ReturnValueOp = Op->Asl.Child;
289         switch (ReturnValueOp->Asl.ParseOpcode)
290         {
291         case PARSEOP_ZERO:
292         case PARSEOP_ONE:
293         case PARSEOP_ONES:
294         case PARSEOP_INTEGER:
295         case PARSEOP_STRING_LITERAL:
296         case PARSEOP_BUFFER:
297         case PARSEOP_PACKAGE:

299             /* Static data return object - check against expected type */

301             ApCheckObjectType (ThisName->Info.Name, ReturnValueOp,
302                 ThisName->Info.ExpectedBtypes, ACPI_NOT_PACKAGE_ELEMENT);

304             /* For packages, check the individual package elements */

306             if (ReturnValueOp->Asl.ParseOpcode == PARSEOP_PACKAGE)
307             {
308                 ApCheckPackage (ReturnValueOp, ThisName);
309             }
310             break;

312         default:
313             /*
314              * All other ops are very difficult or impossible to typecheck at
315              * compile time. These include all Localx, Argx, and method
316              * invocations. Also, NAMESEG and NAMESTRING because the type of
317              * any named object can be changed at runtime (for example,
318              * CopyObject will change the type of the target object.)
319              */
320             break;
321         }
322     }
323 }
```



```
326 /******************************************************************************
327  *
328  * FUNCTION:    ApCheckForPredefinedObject
329  *
330  * PARAMETERS:  Op              - A parse node
331  *              Name            - The ACPI name to be checked
332  *
333  * RETURN:      None
334  *
335  * DESCRIPTION: Check for a predefined name for a static object (created via
336  *              the ASL Name operator). If it is a predefined ACPI name, ensure
337  *              that the name does not require any arguments (which would
338  *              require a control method implemenation of the name), and that
339  *              the type of the object is one of the expected types for the
340  *              predefined name.
341  *
342  *****************************************************************************/

344 void
345 ApCheckForPredefinedObject (
346     ACPI_PARSE_OBJECT       *Op,
347     char                    *Name)
348 {
349     UINT32                      Index;
350     ACPI_PARSE_OBJECT           *ObjectOp;
351     const ACPI_PREDEFINED_INFO  *ThisName;


354     /*
355      * Check for a real predefined name -- not a resource descriptor name
356      * or a predefined scope name
357      */
358     Index = ApCheckForPredefinedName (Op, Name);

360     switch (Index)
361     {
362     case ACPI_NOT_RESERVED_NAME:        /* No underscore or _Txx or _xxx name no
363     case ACPI_PREDEFINED_NAME:          /* Resource Name or reserved scope name
364     case ACPI_COMPILER_RESERVED_NAME:   /* A _Txx that was not emitted by compil

366         /* Nothing to do */
367         return;

369     case ACPI_EVENT_RESERVED_NAME:      /* _Lxx/_Exx/_Wxx/_Qxx methods */

371         /*
372          * These names must be control methods, by definition in ACPI spec.
373          * Also because they are defined to return no value. None of them
374          * require any arguments.
375          */
376         AslError (ASL_ERROR, ASL_MSG_RESERVED_METHOD, Op,
377             "with zero arguments");
378         return;

380     default:

382         break;
383     }

385     /* A standard predefined ACPI name */

387     /*
388      * If this predefined name requires input arguments, then
389      * it must be implemented as a control method
390      */
```

```
391     ThisName = &AcpiGbl_PredefinedMethods[Index];
392     if (METHOD_GET_ARG_COUNT (ThisName->Info.ArgumentList) > 0)
393     {
394         AslError (ASL_ERROR, ASL_MSG_RESERVED_METHOD, Op,
395             "with arguments");
396         return;
397     }

399     /*
400      * If no return value is expected from this predefined name, then
401      * it follows that it must be implemented as a control method
402      * (with zero args, because the args > 0 case was handled above)
403      * Examples are: _DIS, _INI, _IRC, _OFF, _ON, _PSx
404      */
405     if (!ThisName->Info.ExpectedBtypes)
406     {
407         AslError (ASL_ERROR, ASL_MSG_RESERVED_METHOD, Op,
408             "with zero arguments");
409         return;
410     }

412     /* Typecheck the actual object, it is the next argument */

414     ObjectOp = Op->Asl.Child->Asl.Next;
415     ApCheckObjectType (ThisName->Info.Name, Op->Asl.Child->Asl.Next,
416         ThisName->Info.ExpectedBtypes, ACPI_NOT_PACKAGE_ELEMENT);

418     /* For packages, check the individual package elements */

420     if (ObjectOp->Asl.ParseOpcode == PARSEOP_PACKAGE)
421     {
422         ApCheckPackage (ObjectOp, ThisName);
423     }
424 }


427 /******************************************************************************
428  *
429  * FUNCTION:    ApCheckForPredefinedName
430  *
431  * PARAMETERS:  Op              - A parse node
432  *              Name            - NameSeg to check
433  *
434  * RETURN:      None
435  *
436  * DESCRIPTION: Check a NameSeg against the reserved list.
437  *
438  *****************************************************************************/

440 UINT32
441 ApCheckForPredefinedName (
442     ACPI_PARSE_OBJECT       *Op,
443     char                    *Name)
444 {
445     UINT32                      i;
446     const ACPI_PREDEFINED_INFO  *ThisName;


449     if (Name[0] == 0)
450     {
451         AcpiOsPrintf ("Found a null name, external = %s\n",
452             Op->Asl.ExternalName);
453     }

455     /* All reserved names are prefixed with a single underscore */
```



```
457     if (Name[0] != '_')
458     {
459         return (ACPI_NOT_RESERVED_NAME);
460     }

462     /* Check for a standard predefined method name */

464     ThisName = AcpiGbl_PredefinedMethods;
465     for (i = 0; ThisName->Info.Name[0]; i++)
466     {
467         if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
468         {
469             /* Return index into predefined array */
470             return (i);
471         }

473         ThisName++; /* Does not account for extra package data, but is OK */
474     }

476     /* Check for resource names and predefined scope names */

478     ThisName = AcpiGbl_ResourceNames;
479     while (ThisName->Info.Name[0])
480     {
481         if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
482         {
483             return (ACPI_PREDEFINED_NAME);
484         }

486         ThisName++;
487     }

489     ThisName = AcpiGbl_ScopeNames;
490     while (ThisName->Info.Name[0])
491     {
492         if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
493         {
494             return (ACPI_PREDEFINED_NAME);
495         }

497         ThisName++;
498     }

500     /* Check for _Lxx/_Exx/_Wxx/_Qxx/_T_x. Warning if unknown predefined name */

502     return (ApCheckForSpecialName (Op, Name));
503 }


506 /******************************************************************************
507  *
508  * FUNCTION:    ApCheckForSpecialName
509  *
510  * PARAMETERS:  Op              - A parse node
511  *              Name            - NameSeg to check
512  *
513  * RETURN:      None
514  *
515  * DESCRIPTION: Check for the "special" predefined names -
516  *              _Lxx, _Exx, _Qxx, _Wxx, and _T_x
517  *
518  *****************************************************************************/

520 static UINT32
521 ApCheckForSpecialName (
522     ACPI_PARSE_OBJECT       *Op,
```

```
523     char                    *Name)
524 {
525
526     /*
527      * Check for the "special" predefined names. We already know that the
528      * first character is an underscore.
529      *   GPE:  _Lxx
530      *   GPE:  _Exx
531      *   GPE:  _Wxx
532      *   EC:   _Qxx
533      */
534     if ((Name[1] == 'L') ||
535         (Name[1] == 'E') ||
536         (Name[1] == 'W') ||
537         (Name[1] == 'Q'))
538     {
539         /* The next two characters must be hex digits */
540
541         if ((isxdigit ((int) Name[2])) &&
542             (isxdigit ((int) Name[3])))
543         {
544             return (ACPI_EVENT_RESERVED_NAME);
545         }
546     }
547
548     /* Check for the names reserved for the compiler itself: _T_x */
549
550     else if ((Op->Asl.ExternalName[1] == 'T') &&
551              (Op->Asl.ExternalName[2] == '_'))
552     {
553         /* Ignore if actually emitted by the compiler */
554
555         if (Op->Asl.CompileFlags & NODE_COMPILER_EMITTED)
556         {
557             return (ACPI_NOT_RESERVED_NAME);
558         }
559
560         /*
561          * Was not actually emitted by the compiler. This is a special case,
562          * however. If the ASL code being compiled was the result of a
563          * dissasembly, it may possibly contain valid compiler-emitted names
564          * of the form "_T_x". We don't want to issue an error or even a
565          * warning and force the user to manually change the names. So, we
566          * will issue a remark instead.
567          */
568         AslError (ASL_REMARK, ASL_MSG_COMPILER_RESERVED, Op, Op->Asl.ExternalNam
569         return (ACPI_COMPILER_RESERVED_NAME);
570     }
571
572     /*
573      * The name didn't match any of the known predefined names. Flag it as a
574      * warning, since the entire namespace starting with an underscore is
575      * reserved by the ACPI spec.
576      */
577     AslError (ASL_WARNING, ASL_MSG_UNKNOWN_RESERVED_NAME, Op,
578         Op->Asl.ExternalName);
579
580     return (ACPI_NOT_RESERVED_NAME);
581 }
582
583
584 /******************************************************************************
585  *
586  * FUNCTION:    ApCheckObjectType
587  *
588  * PARAMETERS:  PredefinedName  - Name of the predefined object we are checking
```



```
589  *              Op              - Current parse node
590  *              ExpectedBtypes  - Bitmap of expected return type(s)
591  *              PackageIndex    - Index of object within parent package (if
592  *                                applicable - ACPI_NOT_PACKAGE_ELEMENT
593  *                                otherwise)
594  *
595  * RETURN:      None
596  *
597  * DESCRIPTION: Check if the object type is one of the types that is expected
598  *              by the predefined name. Only a limited number of object types
599  *              can be returned by the predefined names.
600  *
601  *****************************************************************************/
602
603 ACPI_STATUS
604 ApCheckObjectType (
605     const char              *PredefinedName,
606     ACPI_PARSE_OBJECT       *Op,
607     UINT32                  ExpectedBtypes,
608     UINT32                  PackageIndex)
609 {
610     UINT32                  ReturnBtype;
611     char                    *TypeName;
612
613
614     if (!Op)
615     {
616         return (AE_TYPE);
617     }
618
619     /* Map the parse opcode to a bitmapped return type (RTYPE) */
620
621     switch (Op->Asl.ParseOpcode)
622     {
623     case PARSEOP_ZERO:
624     case PARSEOP_ONE:
625     case PARSEOP_ONES:
626     case PARSEOP_INTEGER:
627
628         ReturnBtype = ACPI_RTYPE_INTEGER;
629         TypeName = "Integer";
630         break;
631
632     case PARSEOP_STRING_LITERAL:
633
634         ReturnBtype = ACPI_RTYPE_STRING;
635         TypeName = "String";
636         break;
637
638     case PARSEOP_BUFFER:
639
640         ReturnBtype = ACPI_RTYPE_BUFFER;
641         TypeName = "Buffer";
642         break;
643
644     case PARSEOP_PACKAGE:
645     case PARSEOP_VAR_PACKAGE:
646
647         ReturnBtype = ACPI_RTYPE_PACKAGE;
648         TypeName = "Package";
649         break;
650
651     case PARSEOP_NAMESEG:
652     case PARSEOP_NAMESTRING:
653         /*
654          * Ignore any named references within a package object.
```

```
655          *
656          * For Package objects, references are allowed instead of any of the
657          * standard data types (Integer/String/Buffer/Package). These
658          * references are resolved at runtime. NAMESEG and NAMESTRING are
659          * impossible to typecheck at compile time because the type of
660          * any named object can be changed at runtime (for example,
661          * CopyObject will change the type of the target object).
662          */
663         if (PackageIndex != ACPI_NOT_PACKAGE_ELEMENT)
664         {
665             return (AE_OK);
666         }

668         ReturnBtype = ACPI_RTYPE_REFERENCE;
669         TypeName = "Reference";
670         break;

672     default:

674         /* Not one of the supported object types */

676         TypeName = UtGetOpName (Op->Asl.ParseOpcode);
677         goto TypeErrorExit;
678     }

680     /* Exit if the object is one of the expected types */

682     if (ReturnBtype & ExpectedBtypes)
683     {
684         return (AE_OK);
685     }


688 TypeErrorExit:

690     /* Format the expected types and emit an error message */

692     AcpiUtGetExpectedReturnTypes (StringBuffer, ExpectedBtypes);

694     if (PackageIndex == ACPI_NOT_PACKAGE_ELEMENT)
695     {
696         sprintf (MsgBuffer, "%4.4s: found %s, %s required",
697             PredefinedName, TypeName, StringBuffer);
698     }
699     else
700     {
701         sprintf (MsgBuffer, "%4.4s: found %s at index %u, %s required",
702             PredefinedName, TypeName, PackageIndex, StringBuffer);
703     }

705     AslError (ASL_ERROR, ASL_MSG_RESERVED_OPERAND_TYPE, Op, MsgBuffer);
706     return (AE_TYPE);
707 }


710 /*******************************************************************************
711  *
712  * FUNCTION:    ApDisplayReservedNames
713  *
714  * PARAMETERS:  None
715  *
716  * RETURN:      None
717  *
718  * DESCRIPTION: Dump information about the ACPI predefined names and predefined
719  *              resource descriptor names.
720  *
```



```
721  ******************************************************************************/

723 void
724 ApDisplayReservedNames (
725     void)
726 {
727     const ACPI_PREDEFINED_INFO  *ThisName;
728     UINT32                      Count;
729     UINT32                      NumTypes;


732     /*
733      * Predefined names/methods
734      */
735     printf ("\nPredefined Name Information\n\n");

737     Count = 0;
738     ThisName = AcpiGbl_PredefinedMethods;
739     while (ThisName->Info.Name[0])
740     {
741         AcpiUtDisplayPredefinedMethod (MsgBuffer, ThisName, FALSE);
742         Count++;
743         ThisName = AcpiUtGetNextPredefinedMethod (ThisName);
744     }

746     printf ("%u Predefined Names are recognized\n", Count);

748     /*
749      * Resource Descriptor names
750      */
751     printf ("\nPredefined Names for Resource Descriptor Fields\n\n");

753     Count = 0;
754     ThisName = AcpiGbl_ResourceNames;
755     while (ThisName->Info.Name[0])
756     {
757         NumTypes = AcpiUtGetResourceBitWidth (MsgBuffer,
758             ThisName->Info.ArgumentList);

760         printf ("%4.4s    Field is %s bits wide%s\n",
761             ThisName->Info.Name, MsgBuffer,
762             (NumTypes > 1) ? " (depending on descriptor type)" : "");

764         Count++;
765         ThisName++;
766     }

768     printf ("%u Resource Descriptor Field Names are recognized\n", Count);

770     /*
771      * Predefined scope names
772      */
773     printf ("\nPredefined Scope/Device Names (automatically created at root)\n\n

775     ThisName = AcpiGbl_ScopeNames;
776     while (ThisName->Info.Name[0])
777     {
778         printf ("%4.4s    Scope/Device\n", ThisName->Info.Name);
779         ThisName++;
780     }
781 }
```

```
**********************************************************
   21555 Thu Dec 26 13:48:32 2013
new/usr/src/common/acpica/compiler/aslprepkg.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslprepkg - support for ACPI predefined name package objects
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acpredef.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslprepkg")


/* Local prototypes */

static void
ApCheckPackageElements (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *Op,
    UINT8                       Type1,
    UINT32                      Count1,
    UINT8                       Type2,
    UINT32                      Count2);

static void
ApCheckPackageList (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *ParentOp,
    const ACPI_PREDEFINED_INFO  *Package,
    UINT32                      StartIndex,
    UINT32                      Count);

static void
ApPackageTooSmall (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *Op,
    UINT32                      Count,
    UINT32                      ExpectedCount);

static void
ApZeroLengthPackage (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *Op);

static void
ApPackageTooLarge (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *Op,
    UINT32                      Count,
    UINT32                      ExpectedCount);


/******************************************************************************
 *
 * FUNCTION:    ApCheckPackage
 *
 * PARAMETERS:  ParentOp            - Parser op for the package
 *              Predefined          - Pointer to package-specific info for
 *                                    the method
 *
 * RETURN:      None
 *
 * DESCRIPTION: Top-level validation for predefined name return package
 *              objects.
 *
 *****************************************************************************/

void
ApCheckPackage (
    ACPI_PARSE_OBJECT           *ParentOp,
    const ACPI_PREDEFINED_INFO  *Predefined)
{
    ACPI_PARSE_OBJECT           *Op;
    const ACPI_PREDEFINED_INFO  *Package;
    ACPI_STATUS                 Status;
    UINT32                      ExpectedCount;
    UINT32                      Count;
    UINT32                      i;


    /* The package info for this name is in the next table entry */

    Package = Predefined + 1;

    /* First child is the package length */

    Op = ParentOp->Asl.Child;
    Count = (UINT32) Op->Asl.Value.Integer;
```

```
    /*
     * Many of the variable-length top-level packages are allowed to simply
     * have zero elements. This allows the BIOS to tell the host that even
     * though the predefined name/method exists, the feature is not supported.
     * Other package types require one or more elements. In any case, there
     * is no need to continue validation.
     */
    if (!Count)
    {
        switch (Package->RetInfo.Type)
        {
        case ACPI_PTYPE1_FIXED:
        case ACPI_PTYPE1_OPTION:
        case ACPI_PTYPE2_PKG_COUNT:
        case ACPI_PTYPE2_REV_FIXED:

            ApZeroLengthPackage (Predefined->Info.Name, ParentOp);
            break;

        case ACPI_PTYPE1_VAR:
        case ACPI_PTYPE2:
        case ACPI_PTYPE2_COUNT:
        case ACPI_PTYPE2_FIXED:
        case ACPI_PTYPE2_MIN:
        case ACPI_PTYPE2_FIX_VAR:
        default:

            break;
        }

        return;
    }

    /* Get the first element of the package */

    Op = Op->Asl.Next;

    /* Decode the package type */

    switch (Package->RetInfo.Type)
    {
    case ACPI_PTYPE1_FIXED:
        /*
         * The package count is fixed and there are no sub-packages
         *
         * If package is too small, exit.
         * If package is larger than expected, issue warning but continue
         */
        ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
        if (Count < ExpectedCount)
        {
            goto PackageTooSmall;
        }
        else if (Count > ExpectedCount)
        {
            ApPackageTooLarge (Predefined->Info.Name, ParentOp,
                Count, ExpectedCount);
        }

        /* Validate all elements of the package */

        ApCheckPackageElements (Predefined->Info.Name, Op,
            Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
            Package->RetInfo.ObjectType2, Package->RetInfo.Count2);
        break;
```



```
    case ACPI_PTYPE1_VAR:
        /*
         * The package count is variable, there are no sub-packages,
         * and all elements must be of the same type
         */
        for (i = 0; i < Count; i++)
        {
            ApCheckObjectType (Predefined->Info.Name, Op,
                Package->RetInfo.ObjectType1, i);
            Op = Op->Asl.Next;
        }
        break;

    case ACPI_PTYPE1_OPTION:
        /*
         * The package count is variable, there are no sub-packages.
         * There are a fixed number of required elements, and a variable
         * number of optional elements.
         *
         * Check if package is at least as large as the minimum required
         */
        ExpectedCount = Package->RetInfo3.Count;
        if (Count < ExpectedCount)
        {
            goto PackageTooSmall;
        }

        /* Variable number of sub-objects */

        for (i = 0; i < Count; i++)
        {
            if (i < Package->RetInfo3.Count)
            {
                /* These are the required package elements (0, 1, or 2) */

                ApCheckObjectType (Predefined->Info.Name, Op,
                    Package->RetInfo3.ObjectType[i], i);
            }
            else
            {
                /* These are the optional package elements */

                ApCheckObjectType (Predefined->Info.Name, Op,
                    Package->RetInfo3.TailObjectType, i);
            }
            Op = Op->Asl.Next;
        }
        break;

    case ACPI_PTYPE2_REV_FIXED:

        /* First element is the (Integer) revision */

        ApCheckObjectType (Predefined->Info.Name, Op,
            ACPI_RTYPE_INTEGER, 0);

        Op = Op->Asl.Next;
        Count--;

        /* Examine the sub-packages */

        ApCheckPackageList (Predefined->Info.Name, Op,
            Package, 1, Count);
        break;
```

```
260     case ACPI_PTYPE2_PKG_COUNT:

262         /* First element is the (Integer) count of sub-packages to follow */

264         Status = ApCheckObjectType (Predefined->Info.Name, Op,
265             ACPI_RTYPE_INTEGER, 0);

267         /* We must have an integer count from above (otherwise, use Count) */

269         if (ACPI_SUCCESS (Status))
270         {
271             /*
272              * Count cannot be larger than the parent package length, but
273              * allow it to be smaller. The >= accounts for the Integer above.
274              */
275             ExpectedCount = (UINT32) Op->Asl.Value.Integer;
276             if (ExpectedCount >= Count)
277             {
278                 goto PackageTooSmall;
279             }

281             Count = ExpectedCount;
282         }

284         Op = Op->Asl.Next;

286         /* Examine the sub-packages */

288         ApCheckPackageList (Predefined->Info.Name, Op,
289             Package, 1, Count);
290         break;

292     case ACPI_PTYPE2:
293     case ACPI_PTYPE2_FIXED:
294     case ACPI_PTYPE2_MIN:
295     case ACPI_PTYPE2_COUNT:
296     case ACPI_PTYPE2_FIX_VAR:
297         /*
298          * These types all return a single Package that consists of a
299          * variable number of sub-Packages.
300          */

302         /* Examine the sub-packages */

304         ApCheckPackageList (Predefined->Info.Name, Op,
305             Package, 0, Count);
306         break;

308     default:
309         return;
310     }

312     return;

314 PackageTooSmall:
315     ApPackageTooSmall (Predefined->Info.Name, ParentOp,
316         Count, ExpectedCount);
317 }


320 /******************************************************************************
321  *
322  * FUNCTION:    ApCheckPackageElements
323  *
324  * PARAMETERS:  PredefinedName      - Name of the predefined object
325  *              Op                  - Parser op for the package
```



```
326  *              Type1               - Object type for first group
327  *              Count1              - Count for first group
328  *              Type2               - Object type for second group
329  *              Count2              - Count for second group
330  *
331  * RETURN:      None
332  *
333  * DESCRIPTION: Validate all elements of a package. Works with packages that
334  *              are defined to contain up to two groups of different object
335  *              types.
336  *
337  *****************************************************************************/

339 static void
340 ApCheckPackageElements (
341     const char              *PredefinedName,
342     ACPI_PARSE_OBJECT       *Op,
343     UINT8                   Type1,
344     UINT32                  Count1,
345     UINT8                   Type2,
346     UINT32                  Count2)
347 {
348     UINT32                  i;


351     /*
352      * Up to two groups of package elements are supported by the data
353      * structure. All elements in each group must be of the same type.
354      * The second group can have a count of zero.
355      *
356      * Aborts check upon a NULL package element, as this means (at compile
357      * time) that the remainder of the package elements are also NULL
358      * (This is the only way to create NULL package elements.)
359      */
360     for (i = 0; (i < Count1) && Op; i++)
361     {
362         ApCheckObjectType (PredefinedName, Op, Type1, i);
363         Op = Op->Asl.Next;
364     }

366     for (i = 0; (i < Count2) && Op; i++)
367     {
368         ApCheckObjectType (PredefinedName, Op, Type2, (i + Count1));
369         Op = Op->Asl.Next;
370     }
371 }


374 /******************************************************************************
375  *
376  * FUNCTION:    ApCheckPackageList
377  *
378  * PARAMETERS:  PredefinedName      - Name of the predefined object
379  *              ParentOp            - Parser op of the parent package
380  *              Package             - Package info for this predefined name
381  *              StartIndex          - Index in parent package where list begins
382  *              ParentCount         - Element count of parent package
383  *
384  * RETURN:      None
385  *
386  * DESCRIPTION: Validate the individual package elements for a predefined name.
387  *              Handles the cases where the predefined name is defined as a
388  *              Package of Packages (subpackages). These are the types:
389  *
390  *              ACPI_PTYPE2
391  *              ACPI_PTYPE2_FIXED
```

```
 *              ACPI_PTYPE2_MIN
 *              ACPI_PTYPE2_COUNT
 *              ACPI_PTYPE2_FIX_VAR
 *
 *****************************************************************************/

static void
ApCheckPackageList (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *ParentOp,
    const ACPI_PREDEFINED_INFO  *Package,
    UINT32                      StartIndex,
    UINT32                      ParentCount)
{
    ACPI_PARSE_OBJECT           *SubPackageOp = ParentOp;
    ACPI_PARSE_OBJECT           *Op;
    ACPI_STATUS                 Status;
    UINT32                      Count;
    UINT32                      ExpectedCount;
    UINT32                      i;
    UINT32                      j;


    /*
     * Validate each subpackage in the parent Package
     *
     * Note: We ignore NULL package elements on the assumption that
     * they will be initialized by the BIOS or other ASL code.
     */
    for (i = 0; (i < ParentCount) && SubPackageOp; i++)
    {
        /* Each object in the list must be of type Package */

        Status = ApCheckObjectType (PredefinedName, SubPackageOp,
            ACPI_RTYPE_PACKAGE, i + StartIndex);
        if (ACPI_FAILURE (Status))
        {
            goto NextSubpackage;
        }

        /* Examine the different types of expected subpackages */

        Op = SubPackageOp->Asl.Child;

        /* First child is the package length */

        Count = (UINT32) Op->Asl.Value.Integer;
        Op = Op->Asl.Next;

        /* The subpackage must have at least one element */

        if (!Count)
        {
            ApZeroLengthPackage (PredefinedName, SubPackageOp);
            goto NextSubpackage;
        }

        /*
         * Decode the package type.
         * PTYPE2 indicates that a "package of packages" is expected for
         * this name. The various flavors of PTYPE2 indicate the number
         * and format of the subpackages.
         */
        switch (Package->RetInfo.Type)
        {
        case ACPI_PTYPE2:
```



```
        case ACPI_PTYPE2_PKG_COUNT:
        case ACPI_PTYPE2_REV_FIXED:

            /* Each subpackage has a fixed number of elements */

            ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
            if (Count < ExpectedCount)
            {
                ApPackageTooSmall (PredefinedName, SubPackageOp,
                    Count, ExpectedCount);
                break;
            }

            ApCheckPackageElements (PredefinedName, Op,
                Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
                Package->RetInfo.ObjectType2, Package->RetInfo.Count2);
            break;

        case ACPI_PTYPE2_FIX_VAR:
            /*
             * Each subpackage has a fixed number of elements and an
             * optional element
             */
            ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
            if (Count < ExpectedCount)
            {
                ApPackageTooSmall (PredefinedName, SubPackageOp,
                    Count, ExpectedCount);
                break;
            }

            ApCheckPackageElements (PredefinedName, Op,
                Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
                Package->RetInfo.ObjectType2,
                Count - Package->RetInfo.Count1);
            break;

        case ACPI_PTYPE2_FIXED:

            /* Each sub-package has a fixed length */

            ExpectedCount = Package->RetInfo2.Count;
            if (Count < ExpectedCount)
            {
                ApPackageTooSmall (PredefinedName, SubPackageOp,
                    Count, ExpectedCount);
                break;
            }

            /* Check each object/type combination */

            for (j = 0; j < ExpectedCount; j++)
            {
                ApCheckObjectType (PredefinedName, Op,
                    Package->RetInfo2.ObjectType[j], j);

                Op = Op->Asl.Next;
            }
            break;

        case ACPI_PTYPE2_MIN:

            /* Each sub-package has a variable but minimum length */

            ExpectedCount = Package->RetInfo.Count1;
            if (Count < ExpectedCount)
```

```
524             {
525                 ApPackageTooSmall (PredefinedName, SubPackageOp,
526                     Count, ExpectedCount);
527                 break;
528             }
529
530             /* Check the type of each sub-package element */
531
532             ApCheckPackageElements (PredefinedName, Op,
533                 Package->RetInfo.ObjectType1, Count, 0, 0);
534             break;
535
536         case ACPI_PTYPE2_COUNT:
537             /*
538              * First element is the (Integer) count of elements, including
539              * the count field (the ACPI name is NumElements)
540              */
541             Status = ApCheckObjectType (PredefinedName, Op,
542                 ACPI_RTYPE_INTEGER, 0);
543
544             /* We must have an integer count from above (otherwise, use Count) *
545
546             if (ACPI_SUCCESS (Status))
547             {
548                 /*
549                  * Make sure package is large enough for the Count and is
550                  * is as large as the minimum size
551                  */
552                 ExpectedCount = (UINT32) Op->Asl.Value.Integer;
553
554                 if (Count < ExpectedCount)
555                 {
556                     ApPackageTooSmall (PredefinedName, SubPackageOp,
557                         Count, ExpectedCount);
558                     break;
559                 }
560                 else if (Count > ExpectedCount)
561                 {
562                     ApPackageTooLarge (PredefinedName, SubPackageOp,
563                         Count, ExpectedCount);
564                 }
565
566                 /* Some names of this type have a minimum length */
567
568                 if (Count < Package->RetInfo.Count1)
569                 {
570                     ExpectedCount = Package->RetInfo.Count1;
571                     ApPackageTooSmall (PredefinedName, SubPackageOp,
572                         Count, ExpectedCount);
573                     break;
574                 }
575
576                 Count = ExpectedCount;
577             }
578
579             /* Check the type of each sub-package element */
580
581             Op = Op->Asl.Next;
582             ApCheckPackageElements (PredefinedName, Op,
583                 Package->RetInfo.ObjectType1, (Count - 1), 0, 0);
584             break;
585
586         default:
587             break;
588         }
```



```
589
590 NextSubpackage:
591         SubPackageOp = SubPackageOp->Asl.Next;
592     }
593 }
594
595
596 /******************************************************************************
597  *
598  * FUNCTION:    ApPackageTooSmall
599  *
600  * PARAMETERS:  PredefinedName      - Name of the predefined object
601  *              Op                  - Current parser op
602  *              Count               - Actual package element count
603  *              ExpectedCount       - Expected package element count
604  *
605  * RETURN:      None
606  *
607  * DESCRIPTION: Issue error message for a package that is smaller than
608  *              required.
609  *
610  *****************************************************************************/
611
612 static void
613 ApPackageTooSmall (
614     const char                  *PredefinedName,
615     ACPI_PARSE_OBJECT           *Op,
616     UINT32                      Count,
617     UINT32                      ExpectedCount)
618 {
619
620     sprintf (MsgBuffer, "%s: length %u, required minimum is %u",
621         PredefinedName, Count, ExpectedCount);
622
623     AslError (ASL_ERROR, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
624 }
625
626
627 /******************************************************************************
628  *
629  * FUNCTION:    ApZeroLengthPackage
630  *
631  * PARAMETERS:  PredefinedName      - Name of the predefined object
632  *              Op                  - Current parser op
633  *
634  * RETURN:      None
635  *
636  * DESCRIPTION: Issue error message for a zero-length package (a package that
637  *              is required to have a non-zero length). Variable length
638  *              packages seem to be allowed to have zero length, however.
639  *              Even if not allowed, BIOS code does it.
640  *
641  *****************************************************************************/
642
643 static void
644 ApZeroLengthPackage (
645     const char                  *PredefinedName,
646     ACPI_PARSE_OBJECT           *Op)
647 {
648
649     sprintf (MsgBuffer, "%s: length is zero", PredefinedName);
650
651     AslError (ASL_ERROR, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
652 }
653
654
655 /******************************************************************************
```

```
 *
 * FUNCTION:    ApPackageTooLarge
 *
 * PARAMETERS:  PredefinedName      - Name of the predefined object
 *              Op                  - Current parser op
 *              Count               - Actual package element count
 *              ExpectedCount       - Expected package element count
 *
 * RETURN:      None
 *
 * DESCRIPTION: Issue a remark for a package that is larger than expected.
 *
 ******************************************************************************/

static void
ApPackageTooLarge (
    const char                  *PredefinedName,
    ACPI_PARSE_OBJECT           *Op,
    UINT32                      Count,
    UINT32                      ExpectedCount)
{

    sprintf (MsgBuffer, "%s: length is %u, only %u required",
        PredefinedName, Count, ExpectedCount);

    AslError (ASL_REMARK, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
}
```

```
**********************************************************
   32978 Thu Dec 26 13:48:32 2013
new/usr/src/common/acpica/compiler/aslresource.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslresource - Resource template/descriptor utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslresource")


/*******************************************************************************
 *
 * FUNCTION:    RsSmallAddressCheck
 *
 * PARAMETERS:  Minimum             - Address Min value
 *              Maximum             - Address Max value
 *              Length              - Address range value
```



```
 *              Alignment           - Address alignment value
 *              MinOp               - Original Op for Address Min
 *              MaxOp               - Original Op for Address Max
 *              LengthOp            - Original Op for address range
 *              AlignOp             - Original Op for address alignment. If
 *                                    NULL, means "zero value for alignment is
 *                                    OK, and means 64K alignment" (for
 *                                    Memory24 descriptor)
 *              Op                  - Parent Op for entire construct
 *
 * RETURN:      None. Adds error messages to error log if necessary
 *
 * DESCRIPTION: Perform common value checks for "small" address descriptors.
 *              Currently:
 *                  Io, Memory24, Memory32
 *
 *****************************************************************************/

void
RsSmallAddressCheck (
    UINT8                   Type,
    UINT32                  Minimum,
    UINT32                  Maximum,
    UINT32                  Length,
    UINT32                  Alignment,
    ACPI_PARSE_OBJECT       *MinOp,
    ACPI_PARSE_OBJECT       *MaxOp,
    ACPI_PARSE_OBJECT       *LengthOp,
    ACPI_PARSE_OBJECT       *AlignOp,
    ACPI_PARSE_OBJECT       *Op)
{

    if (Gbl_NoResourceChecking)
    {
        return;
    }

    /*
     * Check for a so-called "null descriptor". These are descriptors that are
     * created with most fields set to zero. The intent is that the descriptor
     * will be updated/completed at runtime via a BufferField.
     *
     * If the descriptor does NOT have a resource tag, it cannot be referenced
     * by a BufferField and we will flag this as an error. Conversely, if
     * the descriptor has a resource tag, we will assume that a BufferField
     * will be used to dynamically update it, so no error.
     *
     * A possible enhancement to this check would be to verify that in fact
     * a BufferField is created using the resource tag, and perhaps even
     * verify that a Store is performed to the BufferField.
     *
     * Note: for these descriptors, Alignment is allowed to be zero
     */
    if (!Minimum && !Maximum && !Length)
    {
        if (!Op->Asl.ExternalName)
        {
            /* No resource tag. Descriptor is fixed and is also illegal */

            AslError (ASL_ERROR, ASL_MSG_NULL_DESCRIPTOR, Op, NULL);
        }

        return;
    }

    /* Special case for Memory24, values are compressed */
```

```
128     if (Type == ACPI_RESOURCE_NAME_MEMORY24)
129     {
130         if (!Alignment) /* Alignment==0 means 64K - no invalid alignment */
131         {
132             Alignment = ACPI_UINT16_MAX + 1;
133         }
135         Minimum <<= 8;
136         Maximum <<= 8;
137         Length *= 256;
138     }
140     /* IO descriptor has different definition of min/max, don't check */
142     if (Type != ACPI_RESOURCE_NAME_IO)
143     {
144         /* Basic checks on Min/Max/Length */
146         if (Minimum > Maximum)
147         {
148             AslError (ASL_ERROR, ASL_MSG_INVALID_MIN_MAX, MinOp, NULL);
149         }
150         else if (Length > (Maximum - Minimum + 1))
151         {
152             AslError (ASL_ERROR, ASL_MSG_INVALID_LENGTH, LengthOp, NULL);
153         }
154     }
156     /* Alignment of zero is not in ACPI spec, but is used to mean byte acc */
158     if (!Alignment)
159     {
160         Alignment = 1;
161     }
163     /* Addresses must be an exact multiple of the alignment value */
165     if (Minimum % Alignment)
166     {
167         AslError (ASL_ERROR, ASL_MSG_ALIGNMENT, MinOp, NULL);
168     }
169     if (Maximum % Alignment)
170     {
171         AslError (ASL_ERROR, ASL_MSG_ALIGNMENT, MaxOp, NULL);
172     }
173 }


176 /*******************************************************************************
177  *
178  * FUNCTION:    RsLargeAddressCheck
179  *
180  * PARAMETERS:  Minimum             - Address Min value
181  *              Maximum             - Address Max value
182  *              Length              - Address range value
183  *              Granularity         - Address granularity value
184  *              Flags               - General flags for address descriptors:
185  *                                    _MIF, _MAF, _DEC
186  *              MinOp               - Original Op for Address Min
187  *              MaxOp               - Original Op for Address Max
188  *              LengthOp            - Original Op for address range
189  *              GranOp              - Original Op for address granularity
190  *              Op                  - Parent Op for entire construct
191  *
192  * RETURN:      None. Adds error messages to error log if necessary
```



```
193  *
194  * DESCRIPTION: Perform common value checks for "large" address descriptors.
195  *              Currently:
196  *                  WordIo,     WordBusNumber,  WordSpace
197  *                  DWordIo,    DWordMemory,    DWordSpace
198  *                  QWordIo,    QWordMemory,    QWordSpace
199  *                  ExtendedIo, ExtendedMemory, ExtendedSpace
200  *
201  * _MIF flag set means that the minimum address is fixed and is not relocatable
202  * _MAF flag set means that the maximum address is fixed and is not relocatable
203  * Length of zero means that the record size is variable
204  *
205  * This function implements the LEN/MIF/MAF/MIN/MAX/GRA rules within Table 6-40
206  * of the ACPI 4.0a specification. Added 04/2010.
207  *
208  ******************************************************************************/

210 void
211 RsLargeAddressCheck (
212     UINT64                  Minimum,
213     UINT64                  Maximum,
214     UINT64                  Length,
215     UINT64                  Granularity,
216     UINT8                   Flags,
217     ACPI_PARSE_OBJECT       *MinOp,
218     ACPI_PARSE_OBJECT       *MaxOp,
219     ACPI_PARSE_OBJECT       *LengthOp,
220     ACPI_PARSE_OBJECT       *GranOp,
221     ACPI_PARSE_OBJECT       *Op)
222 {
224     if (Gbl_NoResourceChecking)
225     {
226         return;
227     }
229     /*
230      * Check for a so-called "null descriptor". These are descriptors that are
231      * created with most fields set to zero. The intent is that the descriptor
232      * will be updated/completed at runtime via a BufferField.
233      *
234      * If the descriptor does NOT have a resource tag, it cannot be referenced
235      * by a BufferField and we will flag this as an error. Conversely, if
236      * the descriptor has a resource tag, we will assume that a BufferField
237      * will be used to dynamically update it, so no error.
238      *
239      * A possible enhancement to this check would be to verify that in fact
240      * a BufferField is created using the resource tag, and perhaps even
241      * verify that a Store is performed to the BufferField.
242      */
243     if (!Minimum && !Maximum && !Length && !Granularity)
244     {
245         if (!Op->Asl.ExternalName)
246         {
247             /* No resource tag. Descriptor is fixed and is also illegal */
249             AslError (ASL_ERROR, ASL_MSG_NULL_DESCRIPTOR, Op, NULL);
250         }
252         return;
253     }
255     /* Basic checks on Min/Max/Length */
257     if (Minimum > Maximum)
258     {
```

```
259         AslError (ASL_ERROR, ASL_MSG_INVALID_MIN_MAX, MinOp, NULL);
260         return;
261     }
262     else if (Length > (Maximum - Minimum + 1))
263     {
264         AslError (ASL_ERROR, ASL_MSG_INVALID_LENGTH, LengthOp, NULL);
265         return;
266     }

268     /* If specified (non-zero), ensure granularity is a power-of-two minus one *

270     if (Granularity)
271     {
272         if ((Granularity + 1) &
273              Granularity)
274         {
275             AslError (ASL_ERROR, ASL_MSG_INVALID_GRANULARITY, GranOp, NULL);
276             return;
277         }
278     }

280     /*
281      * Check the various combinations of Length, MinFixed, and MaxFixed
282      */
283     if (Length)
284     {
285         /* Fixed non-zero length */

287         switch (Flags & (ACPI_RESOURCE_FLAG_MIF | ACPI_RESOURCE_FLAG_MAF))
288         {
289         case 0:
290             /*
291              * Fixed length, variable locations (both _MIN and _MAX).
292              * Length must be a multiple of granularity
293              */
294             if (Granularity & Length)
295             {
296                 AslError (ASL_ERROR, ASL_MSG_ALIGNMENT, LengthOp, NULL);
297             }
298             break;

300         case (ACPI_RESOURCE_FLAG_MIF | ACPI_RESOURCE_FLAG_MAF):

302             /* Fixed length, fixed location. Granularity must be zero */

304             if (Granularity != 0)
305             {
306                 AslError (ASL_ERROR, ASL_MSG_INVALID_GRAN_FIXED, GranOp, NULL);
307             }

309             /* Length must be exactly the size of the min/max window */

311             if (Length != (Maximum - Minimum + 1))
312             {
313                 AslError (ASL_ERROR, ASL_MSG_INVALID_LENGTH_FIXED, LengthOp, NUL
314             }
315             break;

317         /* All other combinations are invalid */

319         case ACPI_RESOURCE_FLAG_MIF:
320         case ACPI_RESOURCE_FLAG_MAF:
321         default:

323             AslError (ASL_ERROR, ASL_MSG_INVALID_ADDR_FLAGS, LengthOp, NULL);
324         }
```



```
325     }
326     else
327     {
328         /* Variable length (length==0) */

330         switch (Flags & (ACPI_RESOURCE_FLAG_MIF | ACPI_RESOURCE_FLAG_MAF))
331         {
332         case 0:
333             /*
334              * Both _MIN and _MAX are variable.
335              * No additional requirements, just exit
336              */
337             break;

339         case ACPI_RESOURCE_FLAG_MIF:

341             /* _MIN is fixed. _MIN must be multiple of _GRA */

343             /*
344              * The granularity is defined by the ACPI specification to be a
345              * power-of-two minus one, therefore the granularity is a
346              * bitmask which can be used to easily validate the addresses.
347              */
348             if (Granularity & Minimum)
349             {
350                 AslError (ASL_ERROR, ASL_MSG_ALIGNMENT, MinOp, NULL);
351             }
352             break;

354         case ACPI_RESOURCE_FLAG_MAF:

356             /* _MAX is fixed. (_MAX + 1) must be multiple of _GRA */

358             if (Granularity & (Maximum + 1))
359             {
360                 AslError (ASL_ERROR, ASL_MSG_ALIGNMENT, MaxOp, "-1");
361             }
362             break;

364         /* Both MIF/MAF set is invalid if length is zero */

366         case (ACPI_RESOURCE_FLAG_MIF | ACPI_RESOURCE_FLAG_MAF):
367         default:

369             AslError (ASL_ERROR, ASL_MSG_INVALID_ADDR_FLAGS, LengthOp, NULL);
370         }
371     }
372 }


375 /*******************************************************************************
376  *
377  * FUNCTION:    RsGetStringDataLength
378  *
379  * PARAMETERS:  InitializerOp     - Start of a subtree of init nodes
380  *
381  * RETURN:      Valid string length if a string node is found (otherwise 0)
382  *
383  * DESCRIPTION: In a list of peer nodes, find the first one that contains a
384  *              string and return the length of the string.
385  *
386  ******************************************************************************/

388 UINT16
389 RsGetStringDataLength (
390     ACPI_PARSE_OBJECT       *InitializerOp)
```

```
{

    while (InitializerOp)
    {
        if (InitializerOp->Asl.ParseOpcode == PARSEOP_STRING_LITERAL)
        {
            return ((UINT16) (strlen (InitializerOp->Asl.Value.String) + 1));
        }
        InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
    }

    return (0);
}


/*******************************************************************************
 *
 * FUNCTION:    RsAllocateResourceNode
 *
 * PARAMETERS:  Size        - Size of node in bytes
 *
 * RETURN:      The allocated node - aborts on allocation failure
 *
 * DESCRIPTION: Allocate a resource description node and the resource
 *              descriptor itself (the nodes are used to link descriptors).
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsAllocateResourceNode (
    UINT32                  Size)
{
    ASL_RESOURCE_NODE       *Rnode;


    /* Allocate the node */

    Rnode = UtLocalCalloc (sizeof (ASL_RESOURCE_NODE));

    /* Allocate the resource descriptor itself */

    Rnode->Buffer = UtLocalCalloc (Size);
    Rnode->BufferLength = Size;

    return (Rnode);
}


/*******************************************************************************
 *
 * FUNCTION:    RsCreateResourceField
 *
 * PARAMETERS:  Op              - Resource field node
 *              Name            - Name of the field (Used only to reference
 *                                the field in the ASL, not in the AML)
 *              ByteOffset      - Offset from the field start
 *              BitOffset       - Additional bit offset
 *              BitLength       - Number of bits in the field
 *
 * RETURN:      None, sets fields within the input node
 *
 * DESCRIPTION: Utility function to generate a named bit field within a
 *              resource descriptor. Mark a node as 1) a field in a resource
 *              descriptor, and 2) set the value to be a BIT offset
 *
 ******************************************************************************/
```



```
void
RsCreateResourceField (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name,
    UINT32                  ByteOffset,
    UINT32                  BitOffset,
    UINT32                  BitLength)
{

    Op->Asl.ExternalName = Name;
    Op->Asl.CompileFlags |= NODE_IS_RESOURCE_FIELD;


    Op->Asl.Value.Tag.BitOffset = (ByteOffset * 8) + BitOffset;
    Op->Asl.Value.Tag.BitLength = BitLength;
}


/*******************************************************************************
 *
 * FUNCTION:    RsSetFlagBits
 *
 * PARAMETERS:  *Flags          - Pointer to the flag byte
 *              Op              - Flag initialization node
 *              Position        - Bit position within the flag byte
 *              Default         - Used if the node is DEFAULT.
 *
 * RETURN:      Sets bits within the *Flags output byte.
 *
 * DESCRIPTION: Set a bit in a cumulative flags word from an initialization
 *              node. Will use a default value if the node is DEFAULT, meaning
 *              that no value was specified in the ASL. Used to merge multiple
 *              keywords into a single flags byte.
 *
 ******************************************************************************/

void
RsSetFlagBits (
    UINT8                   *Flags,
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   Position,
    UINT8                   DefaultBit)
{

    if (Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
    {
        /* Use the default bit */

        *Flags |= (DefaultBit << Position);
    }
    else
    {
        /* Use the bit specified in the initialization node */

        *Flags |= (((UINT8) Op->Asl.Value.Integer) << Position);
    }
}


void
RsSetFlagBits16 (
    UINT16                  *Flags,
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   Position,
    UINT8                   DefaultBit)
```

```
523 {
525     if (Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
526     {
527         /* Use the default bit */
529         *Flags |= (DefaultBit << Position);
530     }
531     else
532     {
533         /* Use the bit specified in the initialization node */
535         *Flags |= (((UINT16) Op->Asl.Value.Integer) << Position);
536     }
537 }


540 /*******************************************************************************
541  *
542  * FUNCTION:    RsCompleteNodeAndGetNext
543  *
544  * PARAMETERS:  Op            - Resource node to be completed
545  *
546  * RETURN:      The next peer to the input node.
547  *
548  * DESCRIPTION: Mark the current node completed and return the next peer.
549  *              The node ParseOpcode is set to DEFAULT_ARG, meaning that
550  *              this node is to be ignored from now on.
551  *
552  ******************************************************************************/

554 ACPI_PARSE_OBJECT *
555 RsCompleteNodeAndGetNext (
556     ACPI_PARSE_OBJECT       *Op)
557 {
559     /* Mark this node unused */
561     Op->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
563     /* Move on to the next peer node in the initializer list */
565     return (ASL_GET_PEER_NODE (Op));
566 }


569 /*******************************************************************************
570  *
571  * FUNCTION:    RsCheckListForDuplicates
572  *
573  * PARAMETERS:  Op                  - First op in the initializer list
574  *
575  * RETURN:      None
576  *
577  * DESCRIPTION: Check an initializer list for duplicate values. Emits an error
578  *              if any duplicates are found.
579  *
580  ******************************************************************************/

582 void
583 RsCheckListForDuplicates (
584     ACPI_PARSE_OBJECT       *Op)
585 {
586     ACPI_PARSE_OBJECT       *NextValueOp = Op;
587     ACPI_PARSE_OBJECT       *NextOp;
588     UINT32                  Value;
```



```
591     if (!Op)
592     {
593         return;
594     }
596     /* Search list once for each value in the list */
598     while (NextValueOp)
599     {
600         Value = (UINT32) NextValueOp->Asl.Value.Integer;
602         /* Compare this value to all remaining values in the list */
604         NextOp = ASL_GET_PEER_NODE (NextValueOp);
605         while (NextOp)
606         {
607             if (NextOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
608             {
609                 /* Compare values */
611                 if (Value == (UINT32) NextOp->Asl.Value.Integer)
612                 {
613                     /* Emit error only once per duplicate node */
615                     if (!(NextOp->Asl.CompileFlags & NODE_IS_DUPLICATE))
616                     {
617                         NextOp->Asl.CompileFlags |= NODE_IS_DUPLICATE;
618                         AslError (ASL_ERROR, ASL_MSG_DUPLICATE_ITEM,
619                             NextOp, NULL);
620                     }
621                 }
622             }
624             NextOp = ASL_GET_PEER_NODE (NextOp);
625         }
627         NextValueOp = ASL_GET_PEER_NODE (NextValueOp);
628     }
629 }


632 /*******************************************************************************
633  *
634  * FUNCTION:    RsDoOneResourceDescriptor
635  *
636  * PARAMETERS:  DescriptorTypeOp    - Parent parse node of the descriptor
637  *              CurrentByteOffset   - Offset in the resource descriptor
638  *                                    buffer.
639  *
640  * RETURN:      A valid resource node for the descriptor
641  *
642  * DESCRIPTION: Dispatches the processing of one resource descriptor
643  *
644  ******************************************************************************/

646 ASL_RESOURCE_NODE *
647 RsDoOneResourceDescriptor (
648     ACPI_PARSE_OBJECT       *DescriptorTypeOp,
649     UINT32                  CurrentByteOffset,
650     UINT8                   *State)
651 {
652     ASL_RESOURCE_NODE       *Rnode = NULL;
```

```
655     /* Construct the resource */

657     switch (DescriptorTypeOp->Asl.ParseOpcode)
658     {
659     case PARSEOP_DMA:

661         Rnode = RsDoDmaDescriptor (DescriptorTypeOp,
662                     CurrentByteOffset);
663         break;

665     case PARSEOP_FIXEDDMA:

667         Rnode = RsDoFixedDmaDescriptor (DescriptorTypeOp,
668                     CurrentByteOffset);
669         break;

671     case PARSEOP_DWORDIO:

673         Rnode = RsDoDwordIoDescriptor (DescriptorTypeOp,
674                     CurrentByteOffset);
675         break;

677     case PARSEOP_DWORDMEMORY:

679         Rnode = RsDoDwordMemoryDescriptor (DescriptorTypeOp,
680                     CurrentByteOffset);
681         break;

683     case PARSEOP_DWORDSPACE:

685         Rnode = RsDoDwordSpaceDescriptor (DescriptorTypeOp,
686                     CurrentByteOffset);
687         break;

689     case PARSEOP_ENDDEPENDENTFN:

691         switch (*State)
692         {
693         case ACPI_RSTATE_NORMAL:

695             AslError (ASL_ERROR, ASL_MSG_MISSING_STARTDEPENDENT,
696                 DescriptorTypeOp, NULL);
697             break;

699         case ACPI_RSTATE_START_DEPENDENT:

701             AslError (ASL_ERROR, ASL_MSG_DEPENDENT_NESTING,
702                 DescriptorTypeOp, NULL);
703             break;

705         case ACPI_RSTATE_DEPENDENT_LIST:
706         default:

708             break;
709         }

711         *State = ACPI_RSTATE_NORMAL;
712         Rnode = RsDoEndDependentDescriptor (DescriptorTypeOp,
713                     CurrentByteOffset);
714         break;

716     case PARSEOP_ENDTAG:

718         Rnode = RsDoEndTagDescriptor (DescriptorTypeOp,
719                     CurrentByteOffset);
720         break;
```



```
722     case PARSEOP_EXTENDEDIO:

724         Rnode = RsDoExtendedIoDescriptor (DescriptorTypeOp,
725                     CurrentByteOffset);
726         break;

728     case PARSEOP_EXTENDEDMEMORY:

730         Rnode = RsDoExtendedMemoryDescriptor (DescriptorTypeOp,
731                     CurrentByteOffset);
732         break;

734     case PARSEOP_EXTENDEDSPACE:

736         Rnode = RsDoExtendedSpaceDescriptor (DescriptorTypeOp,
737                     CurrentByteOffset);
738         break;

740     case PARSEOP_FIXEDIO:

742         Rnode = RsDoFixedIoDescriptor (DescriptorTypeOp,
743                     CurrentByteOffset);
744         break;

746     case PARSEOP_INTERRUPT:

748         Rnode = RsDoInterruptDescriptor (DescriptorTypeOp,
749                     CurrentByteOffset);
750         break;

752     case PARSEOP_IO:

754         Rnode = RsDoIoDescriptor (DescriptorTypeOp,
755                     CurrentByteOffset);
756         break;

758     case PARSEOP_IRQ:

760         Rnode = RsDoIrqDescriptor (DescriptorTypeOp,
761                     CurrentByteOffset);
762         break;

764     case PARSEOP_IRQNOFLAGS:

766         Rnode = RsDoIrqNoFlagsDescriptor (DescriptorTypeOp,
767                     CurrentByteOffset);
768         break;

770     case PARSEOP_MEMORY24:

772         Rnode = RsDoMemory24Descriptor (DescriptorTypeOp,
773                     CurrentByteOffset);
774         break;

776     case PARSEOP_MEMORY32:

778         Rnode = RsDoMemory32Descriptor (DescriptorTypeOp,
779                     CurrentByteOffset);
780         break;

782     case PARSEOP_MEMORY32FIXED:

784         Rnode = RsDoMemory32FixedDescriptor (DescriptorTypeOp,
785                     CurrentByteOffset);
786         break;
```

```
788     case PARSEOP_QWORDIO:

790         Rnode = RsDoQwordIoDescriptor (DescriptorTypeOp,
791                     CurrentByteOffset);
792         break;

794     case PARSEOP_QWORDMEMORY:

796         Rnode = RsDoQwordMemoryDescriptor (DescriptorTypeOp,
797                     CurrentByteOffset);
798         break;

800     case PARSEOP_QWORDSPACE:

802         Rnode = RsDoQwordSpaceDescriptor (DescriptorTypeOp,
803                     CurrentByteOffset);
804         break;

806     case PARSEOP_REGISTER:

808         Rnode = RsDoGeneralRegisterDescriptor (DescriptorTypeOp,
809                     CurrentByteOffset);
810         break;

812     case PARSEOP_STARTDEPENDENTFN:

814         switch (*State)
815         {
816         case ACPI_RSTATE_START_DEPENDENT:

818             AslError (ASL_ERROR, ASL_MSG_DEPENDENT_NESTING,
819                 DescriptorTypeOp, NULL);
820             break;

822         case ACPI_RSTATE_NORMAL:
823         case ACPI_RSTATE_DEPENDENT_LIST:
824         default:

826             break;
827         }

829         *State = ACPI_RSTATE_START_DEPENDENT;
830         Rnode = RsDoStartDependentDescriptor (DescriptorTypeOp,
831                     CurrentByteOffset);
832         *State = ACPI_RSTATE_DEPENDENT_LIST;
833         break;

835     case PARSEOP_STARTDEPENDENTFN_NOPRI:

837         switch (*State)
838         {
839         case ACPI_RSTATE_START_DEPENDENT:

841             AslError (ASL_ERROR, ASL_MSG_DEPENDENT_NESTING,
842                 DescriptorTypeOp, NULL);
843             break;

845         case ACPI_RSTATE_NORMAL:
846         case ACPI_RSTATE_DEPENDENT_LIST:
847         default:

849             break;
850         }

852         *State = ACPI_RSTATE_START_DEPENDENT;
```



```
853         Rnode = RsDoStartDependentNoPriDescriptor (DescriptorTypeOp,
854                     CurrentByteOffset);
855         *State = ACPI_RSTATE_DEPENDENT_LIST;
856         break;

858     case PARSEOP_VENDORLONG:

860         Rnode = RsDoVendorLargeDescriptor (DescriptorTypeOp,
861                     CurrentByteOffset);
862         break;

864     case PARSEOP_VENDORSHORT:

866         Rnode = RsDoVendorSmallDescriptor (DescriptorTypeOp,
867                     CurrentByteOffset);
868         break;

870     case PARSEOP_WORDBUSNUMBER:

872         Rnode = RsDoWordBusNumberDescriptor (DescriptorTypeOp,
873                     CurrentByteOffset);
874         break;

876     case PARSEOP_WORDIO:

878         Rnode = RsDoWordIoDescriptor (DescriptorTypeOp,
879                     CurrentByteOffset);
880         break;

882     case PARSEOP_WORDSPACE:

884         Rnode = RsDoWordSpaceDescriptor (DescriptorTypeOp,
885                     CurrentByteOffset);
886         break;

888     case PARSEOP_GPIO_INT:

890         Rnode = RsDoGpioIntDescriptor (DescriptorTypeOp,
891                     CurrentByteOffset);
892         break;

894     case PARSEOP_GPIO_IO:

896         Rnode = RsDoGpioIoDescriptor (DescriptorTypeOp,
897                     CurrentByteOffset);
898         break;

900     case PARSEOP_I2C_SERIALBUS:

902         Rnode = RsDoI2cSerialBusDescriptor (DescriptorTypeOp,
903                     CurrentByteOffset);
904         break;

906     case PARSEOP_SPI_SERIALBUS:

908         Rnode = RsDoSpiSerialBusDescriptor (DescriptorTypeOp,
909                     CurrentByteOffset);
910         break;

912     case PARSEOP_UART_SERIALBUS:

914         Rnode = RsDoUartSerialBusDescriptor (DescriptorTypeOp,
915                     CurrentByteOffset);
916         break;

918     case PARSEOP_DEFAULT_ARG:
```

```
920         /* Just ignore any of these, they are used as fillers/placeholders */
921         break;

923     default:

925         printf ("Unknown resource descriptor type [%s]\n",
926                     DescriptorTypeOp->Asl.ParseOpName);
927         break;
928     }

930     /*
931      * Mark original node as unused, but head of a resource descriptor.
932      * This allows the resource to be installed in the namespace so that
933      * references to the descriptor can be resolved.
934      */
935     DescriptorTypeOp->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
936     DescriptorTypeOp->Asl.CompileFlags = NODE_IS_RESOURCE_DESC;
937     DescriptorTypeOp->Asl.Value.Integer = CurrentByteOffset;

939     if (Rnode)
940     {
941         DescriptorTypeOp->Asl.FinalAmlLength = Rnode->BufferLength;
942         DescriptorTypeOp->Asl.Extra = ((AML_RESOURCE *) Rnode->Buffer)->Descript
943     }

945     return (Rnode);
946 }


949 /*******************************************************************************
950  *
951  * FUNCTION:    RsLinkDescriptorChain
952  *
953  * PARAMETERS:  PreviousRnode       - Pointer to the node that will be previous
954  *                                    to the linked node,  At exit, set to the
955  *                                    last node in the new chain.
956  *              Rnode               - Resource node to link into the list
957  *
958  * RETURN:      Cumulative buffer byte offset of the new segment of chain
959  *
960  * DESCRIPTION: Link a descriptor chain at the end of an existing chain.
961  *
962  ******************************************************************************/

964 UINT32
965 RsLinkDescriptorChain (
966     ASL_RESOURCE_NODE       **PreviousRnode,
967     ASL_RESOURCE_NODE       *Rnode)
968 {
969     ASL_RESOURCE_NODE       *LastRnode;
970     UINT32                  CurrentByteOffset;


973     /* Anything to do? */

975     if (!Rnode)
976     {
977         return (0);
978     }

980     /* Point the previous node to the new node */

982     (*PreviousRnode)->Next = Rnode;
983     CurrentByteOffset = Rnode->BufferLength;
```



```
985     /* Walk to the end of the chain headed by Rnode */

987     LastRnode = Rnode;
988     while (LastRnode->Next)
989     {
990         LastRnode = LastRnode->Next;
991         CurrentByteOffset += LastRnode->BufferLength;
992     }

994     /* Previous node becomes the last node in the chain */

996     *PreviousRnode = LastRnode;
997     return (CurrentByteOffset);
998 }


1001 /*******************************************************************************
1002  *
1003  * FUNCTION:    RsDoResourceTemplate
1004  *
1005  * PARAMETERS:  Op        - Parent of a resource template list
1006  *
1007  * RETURN:      None. Sets input node to point to a list of AML code
1008  *
1009  * DESCRIPTION: Merge a list of resource descriptors into a single AML buffer,
1010  *              in preparation for output to the AML output file.
1011  *
1012  ******************************************************************************/

1014 void
1015 RsDoResourceTemplate (
1016     ACPI_PARSE_OBJECT       *Op)
1017 {
1018     ACPI_PARSE_OBJECT       *BufferLengthOp;
1019     ACPI_PARSE_OBJECT       *BufferOp;
1020     ACPI_PARSE_OBJECT       *DescriptorTypeOp;
1021     ACPI_PARSE_OBJECT       *LastOp = NULL;
1022     UINT32                  CurrentByteOffset = 0;
1023     ASL_RESOURCE_NODE       HeadRnode;
1024     ASL_RESOURCE_NODE       *PreviousRnode;
1025     ASL_RESOURCE_NODE       *Rnode;
1026     UINT8                   State;


1029     /* Mark parent as containing a resource template */

1031     if (Op->Asl.Parent)
1032     {
1033         Op->Asl.Parent->Asl.CompileFlags |= NODE_IS_RESOURCE_DESC;
1034     }

1036     /* ResourceTemplate Opcode is first (Op) */
1037     /* Buffer Length node is first child */

1039     BufferLengthOp = ASL_GET_CHILD_NODE (Op);

1041     /* Buffer Op is first peer */

1043     BufferOp = ASL_GET_PEER_NODE (BufferLengthOp);

1045     /* First Descriptor type is next */

1047     DescriptorTypeOp = ASL_GET_PEER_NODE (BufferOp);

1049     /*
1050      * Process all resource descriptors in the list
```

```
1051      * Note: It is assumed that the EndTag node has been automatically
1052      * inserted at the end of the template by the parser.
1053      */
1054     State = ACPI_RSTATE_NORMAL;
1055     PreviousRnode = &HeadRnode;
1056     while (DescriptorTypeOp)
1057     {
1058         DescriptorTypeOp->Asl.CompileFlags |= NODE_IS_RESOURCE_DESC;
1059         Rnode = RsDoOneResourceDescriptor (DescriptorTypeOp, CurrentByteOffset,
1060                     &State);
1062         /*
1063          * Update current byte offset to indicate the number of bytes from the
1064          * start of the buffer. Buffer can include multiple descriptors, we
1065          * must keep track of the offset of not only each descriptor, but each
1066          * element (field) within each descriptor as well.
1067          */
1068         CurrentByteOffset += RsLinkDescriptorChain (&PreviousRnode, Rnode);
1070         /* Get the next descriptor in the list */
1072         LastOp = DescriptorTypeOp;
1073         DescriptorTypeOp = ASL_GET_PEER_NODE (DescriptorTypeOp);
1074     }
1076     if (State == ACPI_RSTATE_DEPENDENT_LIST)
1077     {
1078         if (LastOp)
1079         {
1080             LastOp = LastOp->Asl.Parent;
1081         }
1082         AslError (ASL_ERROR, ASL_MSG_MISSING_ENDDEPENDENT, LastOp, NULL);
1083     }
1085     /*
1086      * Transform the nodes into the following
1087      *
1088      * Op           -> AML_BUFFER_OP
1089      * First Child  -> BufferLength
1090      * Second Child -> Descriptor Buffer (raw byte data)
1091      */
1092     Op->Asl.ParseOpcode               = PARSEOP_BUFFER;
1093     Op->Asl.AmlOpcode                 = AML_BUFFER_OP;
1094     Op->Asl.CompileFlags              = NODE_AML_PACKAGE | NODE_IS_RESOURCE_DESC
1095     UtSetParseOpName (Op);
1097     BufferLengthOp->Asl.ParseOpcode   = PARSEOP_INTEGER;
1098     BufferLengthOp->Asl.Value.Integer = CurrentByteOffset;
1099     (void) OpcSetOptimalIntegerSize (BufferLengthOp);
1100     UtSetParseOpName (BufferLengthOp);
1102     BufferOp->Asl.ParseOpcode         = PARSEOP_RAW_DATA;
1103     BufferOp->Asl.AmlOpcode           = AML_RAW_DATA_CHAIN;
1104     BufferOp->Asl.AmlOpcodeLength     = 0;
1105     BufferOp->Asl.AmlLength           = CurrentByteOffset;
1106     BufferOp->Asl.Value.Buffer        = (UINT8 *) HeadRnode.Next;
1107     BufferOp->Asl.CompileFlags       |= NODE_IS_RESOURCE_DATA;
1108     UtSetParseOpName (BufferOp);
1110     return;
1111 }
```

```
**********************************************************
   21552 Thu Dec 26 13:48:32 2013
new/usr/src/common/acpica/compiler/aslrestype1.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype1 - Miscellaneous small resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype1")

/*
 * This module contains miscellaneous small resource descriptors:
 *
 * EndTag
 * EndDependentFn
 * Memory24
 * Memory32
 * Memory32Fixed
 * StartDependentFn
 * StartDependentFnNoPri
```



```
 * VendorShort
 */

/******************************************************************************
 *
 * FUNCTION:    RsDoEndTagDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "EndDependentFn" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoEndTagDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ASL_RESOURCE_NODE       *Rnode;


    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_END_TAG));

    Descriptor = Rnode->Buffer;
    Descriptor->EndTag.DescriptorType = ACPI_RESOURCE_NAME_END_TAG |
                                        ASL_RDESC_END_TAG_SIZE;
    Descriptor->EndTag.Checksum = 0;

    return (Rnode);
}


/******************************************************************************
 *
 * FUNCTION:    RsDoEndDependentDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "EndDependentFn" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoEndDependentDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ASL_RESOURCE_NODE       *Rnode;


    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_END_DEPENDENT));

    Descriptor = Rnode->Buffer;
    Descriptor->EndDpf.DescriptorType  = ACPI_RESOURCE_NAME_END_DEPENDENT |
                                         ASL_RDESC_END_DEPEND_SIZE;
    return (Rnode);
```

```
}


/*******************************************************************************
 *
 * FUNCTION:    RsDoMemory24Descriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "Memory24" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoMemory24Descriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_MEMORY24));

    Descriptor = Rnode->Buffer;
    Descriptor->Memory24.DescriptorType  = ACPI_RESOURCE_NAME_MEMORY24;
    Descriptor->Memory24.ResourceLength = 9;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Read/Write type */

            RsSetFlagBits (&Descriptor->Memory24.Flags, InitializerOp, 0, 1);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory24.Flags), 0);
            break;

        case 1: /* Min Address */

            Descriptor->Memory24.Minimum = (UINT16) InitializerOp->Asl.Value.Int
            RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory24.Minimum));
            MinOp = InitializerOp;
            break;

        case 2: /* Max Address */

            Descriptor->Memory24.Maximum = (UINT16) InitializerOp->Asl.Value.Int
            RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory24.Maximum));
            MaxOp = InitializerOp;
            break;
```



```
        case 3: /* Alignment */

            Descriptor->Memory24.Alignment = (UINT16) InitializerOp->Asl.Value.I
            RsCreateWordField (InitializerOp, ACPI_RESTAG_ALIGNMENT,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory24.Alignment));
            break;

        case 4: /* Length */

            Descriptor->Memory24.AddressLength = (UINT16) InitializerOp->Asl.Val
            RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory24.AddressLength))
            LengthOp = InitializerOp;
            break;

        case 5: /* Name */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;

        default:

            AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
            break;
        }

        InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    }

    /* Validate the Min/Max/Len/Align values (Alignment==0 means 64K) */

    RsSmallAddressCheck (ACPI_RESOURCE_NAME_MEMORY24,
        Descriptor->Memory24.Minimum,
        Descriptor->Memory24.Maximum,
        Descriptor->Memory24.AddressLength,
        Descriptor->Memory24.Alignment,
        MinOp, MaxOp, LengthOp, NULL, Op);

    return (Rnode);
}


/*******************************************************************************
 *
 * FUNCTION:    RsDoMemory32Descriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "Memory32" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoMemory32Descriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
```

```
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *AlignOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_MEMORY32));

    Descriptor = Rnode->Buffer;
    Descriptor->Memory32.DescriptorType  = ACPI_RESOURCE_NAME_MEMORY32;
    Descriptor->Memory32.ResourceLength = 17;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Read/Write type */

            RsSetFlagBits (&Descriptor->Memory32.Flags, InitializerOp, 0, 1);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory32.Flags), 0);
            break;

        case 1:  /* Min Address */

            Descriptor->Memory32.Minimum = (UINT32) InitializerOp->Asl.Value.Int
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory32.Minimum));
            MinOp = InitializerOp;
            break;

        case 2: /* Max Address */

            Descriptor->Memory32.Maximum = (UINT32) InitializerOp->Asl.Value.Int
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory32.Maximum));
            MaxOp = InitializerOp;
            break;

        case 3: /* Alignment */

            Descriptor->Memory32.Alignment = (UINT32) InitializerOp->Asl.Value.I
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_ALIGNMENT,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory32.Alignment));
            AlignOp = InitializerOp;
            break;

        case 4: /* Length */

            Descriptor->Memory32.AddressLength = (UINT32) InitializerOp->Asl.Val
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Memory32.AddressLength))
            LengthOp = InitializerOp;
            break;

        case 5: /* Name */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;

        default:

            AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
```



```
            break;
        }

        InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    }

    /* Validate the Min/Max/Len/Align values */

    RsSmallAddressCheck (ACPI_RESOURCE_NAME_MEMORY32,
        Descriptor->Memory32.Minimum,
        Descriptor->Memory32.Maximum,
        Descriptor->Memory32.AddressLength,
        Descriptor->Memory32.Alignment,
        MinOp, MaxOp, LengthOp, AlignOp, Op);

    return (Rnode);
}


/*******************************************************************************
 *
 * FUNCTION:    RsDoMemory32FixedDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "Memory32Fixed" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoMemory32FixedDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_FIXED_MEMORY32));

    Descriptor = Rnode->Buffer;
    Descriptor->FixedMemory32.DescriptorType  = ACPI_RESOURCE_NAME_FIXED_MEMORY3
    Descriptor->FixedMemory32.ResourceLength = 9;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Read/Write type */

            RsSetFlagBits (&Descriptor->FixedMemory32.Flags, InitializerOp, 0, 1
            RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (FixedMemory32.Flags), 0)
            break;

        case 1: /* Address */
```

```
391             Descriptor->FixedMemory32.Address = (UINT32) InitializerOp->Asl.Valu
392             RsCreateDwordField (InitializerOp, ACPI_RESTAG_BASEADDRESS,
393                 CurrentByteOffset + ASL_RESDESC_OFFSET (FixedMemory32.Address));
394             break;

396         case 2: /* Length */

398             Descriptor->FixedMemory32.AddressLength = (UINT32) InitializerOp->As
399             RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
400                 CurrentByteOffset + ASL_RESDESC_OFFSET (FixedMemory32.AddressLen
401             break;

403         case 3: /* Name */

405             UtAttachNamepathToOwner (Op, InitializerOp);
406             break;

408         default:

410             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
411             break;
412         }

414         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
415     }

417     return (Rnode);
418 }


421 /*******************************************************************************
422  *
423  * FUNCTION:    RsDoStartDependentDescriptor
424  *
425  * PARAMETERS:  Op                  - Parent resource descriptor parse node
426  *              CurrentByteOffset   - Offset into the resource template AML
427  *                                    buffer (to track references to the desc)
428  *
429  * RETURN:      Completed resource node
430  *
431  * DESCRIPTION: Construct a short "StartDependentFn" descriptor
432  *
433  ******************************************************************************/

435 ASL_RESOURCE_NODE *
436 RsDoStartDependentDescriptor (
437     ACPI_PARSE_OBJECT       *Op,
438     UINT32                  CurrentByteOffset)
439 {
440     AML_RESOURCE            *Descriptor;
441     ACPI_PARSE_OBJECT       *InitializerOp;
442     ASL_RESOURCE_NODE       *Rnode;
443     ASL_RESOURCE_NODE       *PreviousRnode;
444     ASL_RESOURCE_NODE       *NextRnode;
445     UINT32                  i;
446     UINT8                   State;


449     InitializerOp = Op->Asl.Child;
450     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_START_DEPENDENT));

452     PreviousRnode = Rnode;
453     Descriptor = Rnode->Buffer;

455     /* Increment offset past StartDependent descriptor */
```



```
457     CurrentByteOffset += sizeof (AML_RESOURCE_START_DEPENDENT);

459     /* Descriptor has priority byte */

461     Descriptor->StartDpf.DescriptorType  = ACPI_RESOURCE_NAME_START_DEPENDENT |
462                                         (ASL_RDESC_ST_DEPEND_SIZE + 0x01);

464     /* Process all child initialization nodes */

466     State = ACPI_RSTATE_START_DEPENDENT;
467     for (i = 0; InitializerOp; i++)
468     {
469         switch (i)
470         {
471         case 0: /* Compatibility Priority */

473             if ((UINT8) InitializerOp->Asl.Value.Integer > 2)
474             {
475                 AslError (ASL_ERROR, ASL_MSG_INVALID_PRIORITY,
476                     InitializerOp, NULL);
477             }

479             RsSetFlagBits (&Descriptor->StartDpf.Flags, InitializerOp, 0, 0);
480             break;

482         case 1: /* Performance/Robustness Priority */

484             if ((UINT8) InitializerOp->Asl.Value.Integer > 2)
485             {
486                 AslError (ASL_ERROR, ASL_MSG_INVALID_PERFORMANCE,
487                     InitializerOp, NULL);
488             }

490             RsSetFlagBits (&Descriptor->StartDpf.Flags, InitializerOp, 2, 0);
491             break;

493         default:

495             NextRnode = RsDoOneResourceDescriptor  (InitializerOp,
496                         CurrentByteOffset, &State);

498             /*
499              * Update current byte offset to indicate the number of bytes from t
500              * start of the buffer. Buffer can include multiple descriptors, we
501              * must keep track of the offset of not only each descriptor, but ea
502              * element (field) within each descriptor as well.
503              */
504             CurrentByteOffset += RsLinkDescriptorChain (&PreviousRnode,
505                                      NextRnode);
506             break;
507         }

509         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
510     }

512     return (Rnode);
513 }


516 /*******************************************************************************
517  *
518  * FUNCTION:    RsDoStartDependentNoPriDescriptor
519  *
520  * PARAMETERS:  Op                  - Parent resource descriptor parse node
521  *              CurrentByteOffset   - Offset into the resource template AML
522  *                                    buffer (to track references to the desc)
```

```
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "StartDependentNoPri" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoStartDependentNoPriDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ASL_RESOURCE_NODE       *Rnode;
    ASL_RESOURCE_NODE       *PreviousRnode;
    ASL_RESOURCE_NODE       *NextRnode;
    UINT8                   State;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_START_DEPENDENT_NOPRIO)

    Descriptor = Rnode->Buffer;
    Descriptor->StartDpf.DescriptorType  = ACPI_RESOURCE_NAME_START_DEPENDENT |
                                    ASL_RDESC_ST_DEPEND_SIZE;
    PreviousRnode = Rnode;

    /* Increment offset past StartDependentNoPri descriptor */

    CurrentByteOffset += sizeof (AML_RESOURCE_START_DEPENDENT_NOPRIO);

    /* Process all child initialization nodes */

    State = ACPI_RSTATE_START_DEPENDENT;
    while (InitializerOp)
    {
        NextRnode = RsDoOneResourceDescriptor  (InitializerOp,
                        CurrentByteOffset, &State);

        /*
         * Update current byte offset to indicate the number of bytes from the
         * start of the buffer. Buffer can include multiple descriptors, we
         * must keep track of the offset of not only each descriptor, but each
         * element (field) within each descriptor as well.
         */
        CurrentByteOffset += RsLinkDescriptorChain (&PreviousRnode, NextRnode);

        InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    }

    return (Rnode);
}


/*****************************************************************************
 *
 * FUNCTION:    RsDoVendorSmallDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "VendorShort" descriptor
```



```
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoVendorSmallDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ASL_RESOURCE_NODE       *Rnode;
    UINT8                   *VendorData;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;

    /* Allocate worst case - 7 vendor bytes */

    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_VENDOR_SMALL) + 7);

    Descriptor = Rnode->Buffer;
    Descriptor->VendorSmall.DescriptorType  = ACPI_RESOURCE_NAME_VENDOR_SMALL;
    VendorData = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_SMALL_HEADER);

    /* Process all child initialization nodes */

    InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    for (i = 0; InitializerOp; i++)
    {
        if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
        {
            break;
        }

        /* Maximum 7 vendor data bytes allowed (0-6) */

        if (i >= 7)
        {
            AslError (ASL_ERROR, ASL_MSG_VENDOR_LIST, InitializerOp, NULL);

            /* Eat the excess initializers */

            while (InitializerOp)
            {
                InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
            }
            break;
        }

        VendorData[i] = (UINT8) InitializerOp->Asl.Value.Integer;
        InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    }

    /* Adjust the Rnode buffer size, so correct number of bytes are emitted */

    Rnode->BufferLength -= (7 - i);

    /* Set the length in the Type Tag */

    Descriptor->VendorSmall.DescriptorType |= (UINT8) i;
    return (Rnode);
}
```

```
**********************************************************
   21406 Thu Dec 26 13:48:33 2013
new/usr/src/common/acpica/compiler/aslrestype1i.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype1i - Small I/O-related resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype1i")

/*
 * This module contains the I/O-related small resource descriptors:
 *
 * DMA
 * FixedDMA
 * FixedIO
 * IO
 * IRQ
 * IRQNoFlags
 */
```



```
/******************************************************************************
 *
 * FUNCTION:    RsDoDmaDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "DMA" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoDmaDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;
    UINT8                   DmaChannelMask = 0;
    UINT8                   DmaChannels = 0;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_DMA));

    Descriptor = Rnode->Buffer;
    Descriptor->Dma.DescriptorType  = ACPI_RESOURCE_NAME_DMA |
                                      ASL_RDESC_DMA_SIZE;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* DMA type */

            RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 5, 0);
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_DMATYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Dma.Flags), 5, 2);
            break;

        case 1: /* Bus Master */

            RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_BUSMASTER,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Dma.Flags), 2);
            break;

        case 2: /* Xfer Type (transfer width) */

            RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 0, 0);
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_XFERTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Dma.Flags), 0, 2);
            break;

        case 3: /* Name */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;
```

```
128         default:

130             /* All DMA channel bytes are handled here, after flags and name */

132             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
133             {
134                 /* Up to 8 channels can be specified in the list */

136                 DmaChannels++;
137                 if (DmaChannels > 8)
138                 {
139                     AslError (ASL_ERROR, ASL_MSG_DMA_LIST,
140                         InitializerOp, NULL);
141                     return (Rnode);
142                 }

144                 /* Only DMA channels 0-7 are allowed (mask is 8 bits) */

146                 if (InitializerOp->Asl.Value.Integer > 7)
147                 {
148                     AslError (ASL_ERROR, ASL_MSG_DMA_CHANNEL,
149                         InitializerOp, NULL);
150                 }

152                 /* Build the mask */

154                 DmaChannelMask |=
155                     (1 << ((UINT8) InitializerOp->Asl.Value.Integer));
156             }

158             if (i == 4) /* case 4: First DMA byte */
159             {
160                 /* Check now for duplicates in list */

162                 RsCheckListForDuplicates (InitializerOp);

164                 /* Create a named field at the start of the list */

166                 RsCreateByteField (InitializerOp, ACPI_RESTAG_DMA,
167                     CurrentByteOffset +
168                     ASL_RESDESC_OFFSET (Dma.DmaChannelMask));
169             }
170             break;
171         }

173         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
174     }

176     /* Now we can set the channel mask */

178     Descriptor->Dma.DmaChannelMask = DmaChannelMask;
179     return (Rnode);
180 }


183 /*******************************************************************************
184  *
185  * FUNCTION:    RsDoFixedDmaDescriptor
186  *
187  * PARAMETERS:  Op                  - Parent resource descriptor parse node
188  *              CurrentByteOffset   - Offset into the resource template AML
189  *                                    buffer (to track references to the desc)
190  *
191  * RETURN:      Completed resource node
192  *
```



```
193  * DESCRIPTION: Construct a short "FixedDMA" descriptor
194  *
195  ******************************************************************************/

197 ASL_RESOURCE_NODE *
198 RsDoFixedDmaDescriptor (
199     ACPI_PARSE_OBJECT       *Op,
200     UINT32                  CurrentByteOffset)
201 {
202     AML_RESOURCE            *Descriptor;
203     ACPI_PARSE_OBJECT       *InitializerOp;
204     ASL_RESOURCE_NODE       *Rnode;
205     UINT32                  i;


208     InitializerOp = Op->Asl.Child;
209     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_FIXED_DMA));

211     Descriptor = Rnode->Buffer;
212     Descriptor->FixedDma.DescriptorType =
213         ACPI_RESOURCE_NAME_FIXED_DMA | ASL_RDESC_FIXED_DMA_SIZE;

215     /* Process all child initialization nodes */

217     for (i = 0; InitializerOp; i++)
218     {
219         switch (i)
220         {
221         case 0: /* DMA Request Lines [WORD] (_DMA) */

223             Descriptor->FixedDma.RequestLines = (UINT16) InitializerOp->Asl.Valu
224             RsCreateWordField (InitializerOp, ACPI_RESTAG_DMA,
225                 CurrentByteOffset + ASL_RESDESC_OFFSET (FixedDma.RequestLines));
226             break;

228         case 1: /* DMA Channel [WORD] (_TYP) */

230             Descriptor->FixedDma.Channels = (UINT16) InitializerOp->Asl.Value.In
231             RsCreateWordField (InitializerOp, ACPI_RESTAG_DMATYPE,
232                 CurrentByteOffset + ASL_RESDESC_OFFSET (FixedDma.Channels));
233             break;

235         case 2: /* Transfer Width [BYTE] (_SIZ) */

237             Descriptor->FixedDma.Width = (UINT8) InitializerOp->Asl.Value.Intege
238             RsCreateByteField (InitializerOp, ACPI_RESTAG_XFERTYPE,
239                 CurrentByteOffset + ASL_RESDESC_OFFSET (FixedDma.Width));
240             break;

242         case 3: /* Descriptor Name (optional) */

244             UtAttachNamepathToOwner (Op, InitializerOp);
245             break;

247         default:    /* Ignore any extra nodes */

249             break;
250         }

252         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
253     }

255     return (Rnode);
256 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    RsDoFixedIoDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "FixedIO" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoFixedIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *AddressOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_FIXED_IO));

    Descriptor = Rnode->Buffer;
    Descriptor->Io.DescriptorType  = ACPI_RESOURCE_NAME_FIXED_IO |
                                     ASL_RDESC_FIXED_IO_SIZE;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Base Address */

            Descriptor->FixedIo.Address =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_BASEADDRESS,
                CurrentByteOffset + ASL_RESDESC_OFFSET (FixedIo.Address));
            AddressOp = InitializerOp;
            break;

        case 1: /* Length */

            Descriptor->FixedIo.AddressLength =
                (UINT8) InitializerOp->Asl.Value.Integer;
            RsCreateByteField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (FixedIo.AddressLength));
            break;

        case 2: /* Name */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;

        default:

            AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
            break;
        }
```



```
        InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
    }

    /* Error checks */

    if (Descriptor->FixedIo.Address > 0x03FF)
    {
        AslError (ASL_WARNING, ASL_MSG_ISA_ADDRESS, AddressOp, NULL);
    }

    return (Rnode);
}


/*******************************************************************************
 *
 * FUNCTION:    RsDoIoDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a short "IO" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *AlignOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_IO));

    Descriptor = Rnode->Buffer;
    Descriptor->Io.DescriptorType  = ACPI_RESOURCE_NAME_IO |
                                     ASL_RDESC_IO_SIZE;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Decode size */

            RsSetFlagBits (&Descriptor->Io.Flags, InitializerOp, 0, 1);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Io.Flags), 0);
            break;

        case 1:  /* Min Address */
```

```
391             Descriptor->Io.Minimum =
392                 (UINT16) InitializerOp->Asl.Value.Integer;
393             RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
394                 CurrentByteOffset + ASL_RESDESC_OFFSET (Io.Minimum));
395             MinOp = InitializerOp;
396             break;

398         case 2: /* Max Address */

400             Descriptor->Io.Maximum =
401                 (UINT16) InitializerOp->Asl.Value.Integer;
402             RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
403                 CurrentByteOffset + ASL_RESDESC_OFFSET (Io.Maximum));
404             MaxOp = InitializerOp;
405             break;

407         case 3: /* Alignment */

409             Descriptor->Io.Alignment =
410                 (UINT8) InitializerOp->Asl.Value.Integer;
411             RsCreateByteField (InitializerOp, ACPI_RESTAG_ALIGNMENT,
412                 CurrentByteOffset + ASL_RESDESC_OFFSET (Io.Alignment));
413             AlignOp = InitializerOp;
414             break;

416         case 4: /* Length */

418             Descriptor->Io.AddressLength =
419                 (UINT8) InitializerOp->Asl.Value.Integer;
420             RsCreateByteField (InitializerOp, ACPI_RESTAG_LENGTH,
421                 CurrentByteOffset + ASL_RESDESC_OFFSET (Io.AddressLength));
422             LengthOp = InitializerOp;
423             break;

425         case 5: /* Name */

427             UtAttachNamepathToOwner (Op, InitializerOp);
428             break;

430         default:

432             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
433             break;
434         }

436         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
437     }

439     /* Validate the Min/Max/Len/Align values */

441     RsSmallAddressCheck (ACPI_RESOURCE_NAME_IO,
442         Descriptor->Io.Minimum,
443         Descriptor->Io.Maximum,
444         Descriptor->Io.AddressLength,
445         Descriptor->Io.Alignment,
446         MinOp, MaxOp, LengthOp, AlignOp, Op);

448     return (Rnode);
449 }


452 /*******************************************************************************
453  *
454  * FUNCTION:    RsDoIrqDescriptor
455  *
456  * PARAMETERS:  Op                  - Parent resource descriptor parse node
```



```
457  *              CurrentByteOffset   - Offset into the resource template AML
458  *                                    buffer (to track references to the desc)
459  *
460  * RETURN:      Completed resource node
461  *
462  * DESCRIPTION: Construct a short "IRQ" descriptor
463  *
464  ******************************************************************************/

466 ASL_RESOURCE_NODE *
467 RsDoIrqDescriptor (
468     ACPI_PARSE_OBJECT       *Op,
469     UINT32                  CurrentByteOffset)
470 {
471     AML_RESOURCE            *Descriptor;
472     ACPI_PARSE_OBJECT       *InitializerOp;
473     ASL_RESOURCE_NODE       *Rnode;
474     UINT32                  Interrupts = 0;
475     UINT16                  IrqMask = 0;
476     UINT32                  i;


479     InitializerOp = Op->Asl.Child;
480     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_IRQ));

482     /* Length = 3 (with flag byte) */

484     Descriptor = Rnode->Buffer;
485     Descriptor->Irq.DescriptorType  = ACPI_RESOURCE_NAME_IRQ |
486                                       (ASL_RDESC_IRQ_SIZE + 0x01);

488     /* Process all child initialization nodes */

490     for (i = 0; InitializerOp; i++)
491     {
492         switch (i)
493         {
494         case 0: /* Interrupt Type (or Mode - edge/level) */

496             RsSetFlagBits (&Descriptor->Irq.Flags, InitializerOp, 0, 1);
497             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTTYPE,
498                 CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.Flags), 0);
499             break;

501         case 1: /* Interrupt Level (or Polarity - Active high/low) */

503             RsSetFlagBits (&Descriptor->Irq.Flags, InitializerOp, 3, 0);
504             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTLEVEL,
505                 CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.Flags), 3);
506             break;

508         case 2: /* Share Type - Default: exclusive (0) */

510             RsSetFlagBits (&Descriptor->Irq.Flags, InitializerOp, 4, 0);
511             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTSHARE,
512                 CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.Flags), 4);
513             break;

515         case 3: /* Name */

517             UtAttachNamepathToOwner (Op, InitializerOp);
518             break;

520         default:

522             /* All IRQ bytes are handled here, after the flags and name */
```

```
524             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
525             {
526                 /* Up to 16 interrupts can be specified in the list */

528                 Interrupts++;
529                 if (Interrupts > 16)
530                 {
531                     AslError (ASL_ERROR, ASL_MSG_INTERRUPT_LIST,
532                         InitializerOp, NULL);
533                     return (Rnode);
534                 }

536                 /* Only interrupts 0-15 are allowed (mask is 16 bits) */

538                 if (InitializerOp->Asl.Value.Integer > 15)
539                 {
540                     AslError (ASL_ERROR, ASL_MSG_INTERRUPT_NUMBER,
541                         InitializerOp, NULL);
542                 }
543                 else
544                 {
545                     IrqMask |= (1 << (UINT8) InitializerOp->Asl.Value.Integer);
546                 }
547             }

549             /* Case 4: First IRQ value in list */

551             if (i == 4)
552             {
553                 /* Check now for duplicates in list */

555                 RsCheckListForDuplicates (InitializerOp);

557                 /* Create a named field at the start of the list */

559                 RsCreateWordField (InitializerOp, ACPI_RESTAG_INTERRUPT,
560                     CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.IrqMask));
561             }
562             break;
563         }

565         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
566     }

568     /* Now we can set the channel mask */

570     Descriptor->Irq.IrqMask = IrqMask;
571     return (Rnode);
572 }


575 /*******************************************************************************
576  *
577  * FUNCTION:    RsDoIrqNoFlagsDescriptor
578  *
579  * PARAMETERS:  Op                  - Parent resource descriptor parse node
580  *              CurrentByteOffset   - Offset into the resource template AML
581  *                                    buffer (to track references to the desc)
582  *
583  * RETURN:      Completed resource node
584  *
585  * DESCRIPTION: Construct a short "IRQNoFlags" descriptor
586  *
587  ******************************************************************************/
```



```
589 ASL_RESOURCE_NODE *
590 RsDoIrqNoFlagsDescriptor (
591     ACPI_PARSE_OBJECT       *Op,
592     UINT32                  CurrentByteOffset)
593 {
594     AML_RESOURCE            *Descriptor;
595     ACPI_PARSE_OBJECT       *InitializerOp;
596     ASL_RESOURCE_NODE       *Rnode;
597     UINT16                  IrqMask = 0;
598     UINT32                  Interrupts = 0;
599     UINT32                  i;


602     InitializerOp = Op->Asl.Child;
603     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_IRQ_NOFLAGS));

605     Descriptor = Rnode->Buffer;
606     Descriptor->Irq.DescriptorType  = ACPI_RESOURCE_NAME_IRQ |
607                                       ASL_RDESC_IRQ_SIZE;

609     /* Process all child initialization nodes */

611     for (i = 0; InitializerOp; i++)
612     {
613         switch (i)
614         {
615         case 0: /* Name */

617             UtAttachNamepathToOwner (Op, InitializerOp);
618             break;

620         default:

622             /* IRQ bytes are handled here, after the flags and name */

624             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
625             {
626                 /* Up to 16 interrupts can be specified in the list */

628                 Interrupts++;
629                 if (Interrupts > 16)
630                 {
631                     AslError (ASL_ERROR, ASL_MSG_INTERRUPT_LIST,
632                         InitializerOp, NULL);
633                     return (Rnode);
634                 }

636                 /* Only interrupts 0-15 are allowed (mask is 16 bits) */

638                 if (InitializerOp->Asl.Value.Integer > 15)
639                 {
640                     AslError (ASL_ERROR, ASL_MSG_INTERRUPT_NUMBER,
641                         InitializerOp, NULL);
642                 }
643                 else
644                 {
645                     IrqMask |= (1 << ((UINT8) InitializerOp->Asl.Value.Integer))
646                 }
647             }

649             /* Case 1: First IRQ value in list */

651             if (i == 1)
652             {
653                 /* Check now for duplicates in list */
```

```
655                 RsCheckListForDuplicates (InitializerOp);
657                 /* Create a named field at the start of the list */
659                 RsCreateWordField (InitializerOp, ACPI_RESTAG_INTERRUPT,
660                     CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.IrqMask));
661             }
662             break;
663         }
665         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
666     }
668     /* Now we can set the interrupt mask */
670     Descriptor->Irq.IrqMask = IrqMask;
671     return (Rnode);
672 }
```

```
**********************************************************
   15490 Thu Dec 26 13:48:33 2013
new/usr/src/common/acpica/compiler/aslrestype2.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2 - Miscellaneous Large resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2")

/*
 * This module contains miscellaneous large resource descriptors:
 *
 * Register
 * Interrupt
 * VendorLong
 */

/******************************************************************************
```



```
 *
 * FUNCTION:    RsDoGeneralRegisterDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "Register" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoGeneralRegisterDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ASL_RESOURCE_NODE       *Rnode;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_GENERIC_REGISTER));

    Descriptor = Rnode->Buffer;
    Descriptor->GenericReg.DescriptorType = ACPI_RESOURCE_NAME_GENERIC_REGISTER;
    Descriptor->GenericReg.ResourceLength = 12;

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Address space */

            Descriptor->GenericReg.AddressSpaceId = (UINT8) InitializerOp->Asl.V
            RsCreateByteField (InitializerOp, ACPI_RESTAG_ADDRESSSPACE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (GenericReg.AddressSpaceI
           break;

        case 1: /* Register Bit Width */

            Descriptor->GenericReg.BitWidth = (UINT8) InitializerOp->Asl.Value.I
            RsCreateByteField (InitializerOp, ACPI_RESTAG_REGISTERBITWIDTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (GenericReg.BitWidth));
            break;

        case 2: /* Register Bit Offset */

            Descriptor->GenericReg.BitOffset = (UINT8) InitializerOp->Asl.Value.
            RsCreateByteField (InitializerOp, ACPI_RESTAG_REGISTERBITOFFSET,
                CurrentByteOffset + ASL_RESDESC_OFFSET (GenericReg.BitOffset));
            break;

        case 3: /* Register Address */

            Descriptor->GenericReg.Address = InitializerOp->Asl.Value.Integer;
            RsCreateQwordField (InitializerOp, ACPI_RESTAG_ADDRESS,
                CurrentByteOffset + ASL_RESDESC_OFFSET (GenericReg.Address));
            break;

        case 4: /* Access Size (ACPI 3.0) */
```

```
128             Descriptor->GenericReg.AccessSize = (UINT8) InitializerOp->Asl.Value
129             RsCreateByteField (InitializerOp, ACPI_RESTAG_ACCESSSIZE,
130                 CurrentByteOffset + ASL_RESDESC_OFFSET (GenericReg.AccessSize));
131
132             if (Descriptor->GenericReg.AccessSize > AML_FIELD_ACCESS_QWORD)
133             {
134                 AslError (ASL_ERROR, ASL_MSG_INVALID_ACCESS_SIZE,
135                     InitializerOp, NULL);
136             }
137             break;
138
139         case 5: /* ResourceTag (ACPI 3.0b) */
140
141             UtAttachNamepathToOwner (Op, InitializerOp);
142             break;
143
144         default:
145
146             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
147             break;
148         }
149
150         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
151     }
152     return (Rnode);
153 }
154
155
156 /*******************************************************************************
157  *
158  * FUNCTION:    RsDoInterruptDescriptor
159  *
160  * PARAMETERS:  Op                  - Parent resource descriptor parse node
161  *              CurrentByteOffset   - Offset into the resource template AML
162  *                                    buffer (to track references to the desc)
163  *
164  * RETURN:      Completed resource node
165  *
166  * DESCRIPTION: Construct a long "Interrupt" descriptor
167  *
168  ******************************************************************************/
169
170 ASL_RESOURCE_NODE *
171 RsDoInterruptDescriptor (
172     ACPI_PARSE_OBJECT       *Op,
173     UINT32                  CurrentByteOffset)
174 {
175     AML_RESOURCE            *Descriptor;
176     AML_RESOURCE            *Rover = NULL;
177     ACPI_PARSE_OBJECT       *InitializerOp;
178     ASL_RESOURCE_NODE       *Rnode;
179     UINT16                  StringLength = 0;
180     UINT32                  OptionIndex = 0;
181     UINT32                  i;
182     BOOLEAN                 HasResSourceIndex = FALSE;
183     UINT8                   ResSourceIndex = 0;
184     UINT8                   *ResSourceString = NULL;
185
186
187     InitializerOp = Op->Asl.Child;
188     StringLength = RsGetStringDataLength (InitializerOp);
189
190     /* Count the interrupt numbers */
191
192     for (i = 0; InitializerOp; i++)
```



```
193     {
194         InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
195
196         if (i <= 6)
197         {
198             if (i == 3 &&
199                 InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
200             {
201                 /*
202                  * ResourceSourceIndex was specified, always make room for
203                  * it, even if the ResourceSource was omitted.
204                  */
205                 OptionIndex++;
206             }
207
208             continue;
209         }
210
211         OptionIndex += 4;
212     }
213
214     InitializerOp = Op->Asl.Child;
215     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_EXTENDED_IRQ) +
216                 1 + OptionIndex + StringLength);
217
218     Descriptor = Rnode->Buffer;
219     Descriptor->ExtendedIrq.DescriptorType  = ACPI_RESOURCE_NAME_EXTENDED_IRQ;
220
221     /*
222      * Initial descriptor length -- may be enlarged if there are
223      * optional fields present
224      */
225     Descriptor->ExtendedIrq.ResourceLength  = 2;  /* Flags and table length byte
226     Descriptor->ExtendedIrq.InterruptCount  = 0;
227
228     Rover = ACPI_CAST_PTR (AML_RESOURCE,
229                 (&(Descriptor->ExtendedIrq.Interrupts[0])));
230
231     /* Process all child initialization nodes */
232
233     for (i = 0; InitializerOp; i++)
234     {
235         switch (i)
236         {
237         case 0: /* Resource Usage (Default: consumer (1) */
238
239             RsSetFlagBits (&Descriptor->ExtendedIrq.Flags, InitializerOp, 0, 1);
240             break;
241
242         case 1: /* Interrupt Type (or Mode - edge/level) */
243
244             RsSetFlagBits (&Descriptor->ExtendedIrq.Flags, InitializerOp, 1, 0);
245             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTTYPE,
246                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtendedIrq.Flags), 1);
247             break;
248
249         case 2: /* Interrupt Level (or Polarity - Active high/low) */
250
251             RsSetFlagBits (&Descriptor->ExtendedIrq.Flags, InitializerOp, 2, 0);
252             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTLEVEL,
253                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtendedIrq.Flags), 2);
254             break;
255
256         case 3: /* Share Type - Default: exclusive (0) */
257
258             RsSetFlagBits (&Descriptor->ExtendedIrq.Flags, InitializerOp, 3, 0);
```

```
259             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTSHARE,
260                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtendedIrq.Flags), 3);
261             break;

263         case 4: /* ResSourceIndex [Optional Field - BYTE] */

265             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
266             {
267                 HasResSourceIndex = TRUE;
268                 ResSourceIndex = (UINT8) InitializerOp->Asl.Value.Integer;
269             }
270             break;

272         case 5: /* ResSource [Optional Field - STRING] */

274             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
275                 (InitializerOp->Asl.Value.String))
276             {
277                 if (StringLength)
278                 {
279                     ResSourceString = (UINT8 *) InitializerOp->Asl.Value.String;
280                 }

282                 /* ResourceSourceIndex must also be valid */

284                 if (!HasResSourceIndex)
285                 {
286                     AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
287                         InitializerOp, NULL);
288                 }
289             }

291 #if 0
292             /*
293              * Not a valid ResourceSource, ResourceSourceIndex must also
294              * be invalid
295              */
296             else if (HasResSourceIndex)
297             {
298                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
299                     InitializerOp, NULL);
300             }
301 #endif
302             break;

304         case 6: /* ResourceTag */

306             UtAttachNamepathToOwner (Op, InitializerOp);
307             break;

309         default:
310             /*
311              * Interrupt Numbers come through here, repeatedly
312              */

314             /* Maximum 255 interrupts allowed for this descriptor */

316             if (Descriptor->ExtendedIrq.InterruptCount == 255)
317             {
318                 AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_LIST,
319                     InitializerOp, NULL);
320                 return (Rnode);
321             }

323             /* Each interrupt number must be a 32-bit value */
```



```
325             if (InitializerOp->Asl.Value.Integer > ACPI_UINT32_MAX)
326             {
327                 AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_NUMBER,
328                     InitializerOp, NULL);
329             }

331             /* Save the integer and move pointer to the next one */

333             Rover->DwordItem = (UINT32) InitializerOp->Asl.Value.Integer;
334             Rover = ACPI_ADD_PTR (AML_RESOURCE, &(Rover->DwordItem), 4);
335             Descriptor->ExtendedIrq.InterruptCount++;
336             Descriptor->ExtendedIrq.ResourceLength += 4;

338             /* Case 7: First interrupt number in list */

340             if (i == 7)
341             {
342                 if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
343                 {
344                     /* Must be at least one interrupt */

346                     AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_LIST_MIN,
347                         InitializerOp, NULL);
348                 }

350                 /* Check now for duplicates in list */

352                 RsCheckListForDuplicates (InitializerOp);

354                 /* Create a named field at the start of the list */

356                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_INTERRUPT,
357                     CurrentByteOffset +
358                     ASL_RESDESC_OFFSET (ExtendedIrq.Interrupts[0]));
359             }
360         }

362         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
363     }


366     /* Add optional ResSourceIndex if present */

368     if (HasResSourceIndex)
369     {
370         Rover->ByteItem = ResSourceIndex;
371         Rover = ACPI_ADD_PTR (AML_RESOURCE, &(Rover->ByteItem), 1);
372         Descriptor->ExtendedIrq.ResourceLength += 1;
373     }

375     /* Add optional ResSource string if present */

377     if (StringLength && ResSourceString)
378     {

380         strcpy ((char *) Rover, (char *) ResSourceString);
381         Rover = ACPI_ADD_PTR (
382                     AML_RESOURCE, &(Rover->ByteItem), StringLength);

384         Descriptor->ExtendedIrq.ResourceLength = (UINT16)
385             (Descriptor->ExtendedIrq.ResourceLength + StringLength);
386     }

388     Rnode->BufferLength = (ASL_RESDESC_OFFSET (ExtendedIrq.Interrupts[0]) -
389                            ASL_RESDESC_OFFSET (ExtendedIrq.DescriptorType))
390                            + OptionIndex + StringLength;
```

```
391     return (Rnode);
392 }


395 /*******************************************************************************
396  *
397  * FUNCTION:    RsDoVendorLargeDescriptor
398  *
399  * PARAMETERS:  Op                  - Parent resource descriptor parse node
400  *              CurrentByteOffset   - Offset into the resource template AML
401  *                                    buffer (to track references to the desc)
402  *
403  * RETURN:      Completed resource node
404  *
405  * DESCRIPTION: Construct a long "VendorLong" descriptor
406  *
407  ******************************************************************************/

409 ASL_RESOURCE_NODE *
410 RsDoVendorLargeDescriptor (
411     ACPI_PARSE_OBJECT       *Op,
412     UINT32                  CurrentByteOffset)
413 {
414     AML_RESOURCE            *Descriptor;
415     ACPI_PARSE_OBJECT       *InitializerOp;
416     ASL_RESOURCE_NODE       *Rnode;
417     UINT8                   *VendorData;
418     UINT32                  i;


421     /* Count the number of data bytes */

423     InitializerOp = Op->Asl.Child;
424     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);

426     for (i = 0; InitializerOp; i++)
427     {
428         if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
429         {
430             break;
431         }
432         InitializerOp = InitializerOp->Asl.Next;
433     }

435     InitializerOp = Op->Asl.Child;
436     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
437     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_VENDOR_LARGE) + i);

439     Descriptor = Rnode->Buffer;
440     Descriptor->VendorLarge.DescriptorType  = ACPI_RESOURCE_NAME_VENDOR_LARGE;
441     Descriptor->VendorLarge.ResourceLength = (UINT16) i;

443     /* Point to end-of-descriptor for vendor data */

445     VendorData = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_LARGE_HEADER);

447     /* Process all child initialization nodes */

449     for (i = 0; InitializerOp; i++)
450     {
451         if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
452         {
453             break;
454         }

456         VendorData[i] = (UINT8) InitializerOp->Asl.Value.Integer;
```



```
457         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
458     }

460     return (Rnode);
461 }
```

```
**********************************************************
   26143 Thu Dec 26 13:48:33 2013
new/usr/src/common/acpica/compiler/aslrestype2d.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2d - Large DWord address resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2d")

/*
 * This module contains the Dword (32-bit) address space descriptors:
 *
 * DwordIO
 * DwordMemory
 * DwordSpace
 */

/******************************************************************************
 *
```



```
 * FUNCTION:    RsDoDwordIoDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "DwordIO" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoDwordIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *GranOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT16                  StringLength = 0;
    UINT32                  OptionIndex = 0;
    UINT8                   *OptionalFields;
    UINT32                  i;
    BOOLEAN                 ResSourceIndex = FALSE;


    InitializerOp = Op->Asl.Child;
    StringLength = RsGetStringDataLength (InitializerOp);

    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;
    Descriptor->Address32.ResourceType   = ACPI_ADDRESS_TYPE_IO_RANGE;

    /*
     * Initial descriptor length -- may be enlarged if there are
     * optional fields present
     */
    OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS32);
    Descriptor->Address32.ResourceLength = (UINT16)
        (sizeof (AML_RESOURCE_ADDRESS32) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Usage */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
            break;

        case 1: /* MinType */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 2);
```

```
127             break;
129         case 2: /* MaxType */
131             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 3, 0);
132             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
133                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 3);
134             break;
136         case 3: /* DecodeType */
138             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 1, 0);
139             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
140                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 1);
141             break;
143         case 4: /* Range Type */
145             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
146             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
147                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
148             break;
150         case 5: /* Address Granularity */
152             Descriptor->Address32.Granularity =
153                 (UINT32) InitializerOp->Asl.Value.Integer;
154             RsCreateDwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
155                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Granularity));
156             GranOp = InitializerOp;
157             break;
159         case 6: /* Address Min */
161             Descriptor->Address32.Minimum =
162                 (UINT32) InitializerOp->Asl.Value.Integer;
163             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
164                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Minimum));
165             MinOp = InitializerOp;
166             break;
168         case 7: /* Address Max */
170             Descriptor->Address32.Maximum =
171                 (UINT32) InitializerOp->Asl.Value.Integer;
172             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
173                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Maximum));
174             MaxOp = InitializerOp;
175             break;
177         case 8: /* Translation Offset */
179             Descriptor->Address32.TranslationOffset =
180                 (UINT32) InitializerOp->Asl.Value.Integer;
181             RsCreateDwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
182                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.TranslationOff
183             break;
185         case 9: /* Address Length */
187             Descriptor->Address32.AddressLength =
188                 (UINT32) InitializerOp->Asl.Value.Integer;
189             RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
190                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.AddressLength)
191             LengthOp = InitializerOp;
192             break;
```



```
194         case 10: /* ResSourceIndex [Optional Field - BYTE] */
196             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
197             {
198                 /* Found a valid ResourceSourceIndex */
200                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
201                 OptionIndex++;
202                 Descriptor->Address32.ResourceLength++;
203                 ResSourceIndex = TRUE;
204             }
205             break;
207         case 11: /* ResSource [Optional Field - STRING] */
209             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
210                 (InitializerOp->Asl.Value.String))
211             {
212                 if (StringLength)
213                 {
214                     /* Found a valid ResourceSource */
216                     Descriptor->Address32.ResourceLength = (UINT16)
217                         (Descriptor->Address32.ResourceLength + StringLength);
219                     strcpy ((char *)
220                         &OptionalFields[OptionIndex],
221                         InitializerOp->Asl.Value.String);
223                     /* ResourceSourceIndex must also be valid */
225                     if (!ResSourceIndex)
226                     {
227                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
228                             InitializerOp, NULL);
229                     }
230                 }
231             }
233 #if 0
234             /*
235              * Not a valid ResourceSource, ResourceSourceIndex must also
236              * be invalid
237              */
238             else if (ResSourceIndex)
239             {
240                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
241                     InitializerOp, NULL);
242             }
243 #endif
244             break;
246         case 12: /* ResourceTag */
248             UtAttachNamepathToOwner (Op, InitializerOp);
249             break;
251         case 13: /* Type */
253             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
254             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
255                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
256             break;
258         case 14: /* Translation Type */
```

```
260             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
261             RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
262                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
263             break;

265         default:

267             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
268             break;
269         }

271         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
272     }

274     /* Validate the Min/Max/Len/Gran values */

276     RsLargeAddressCheck (
277         (UINT64) Descriptor->Address32.Minimum,
278         (UINT64) Descriptor->Address32.Maximum,
279         (UINT64) Descriptor->Address32.AddressLength,
280         (UINT64) Descriptor->Address32.Granularity,
281         Descriptor->Address32.Flags,
282         MinOp, MaxOp, LengthOp, GranOp, Op);

284     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS32) +
285                           OptionIndex + StringLength;
286     return (Rnode);
287 }


290 /*******************************************************************************
291  *
292  * FUNCTION:    RsDoDwordMemoryDescriptor
293  *
294  * PARAMETERS:  Op                  - Parent resource descriptor parse node
295  *              CurrentByteOffset   - Offset into the resource template AML
296  *                                    buffer (to track references to the desc)
297  *
298  * RETURN:      Completed resource node
299  *
300  * DESCRIPTION: Construct a long "DwordMemory" descriptor
301  *
302  ******************************************************************************/

304 ASL_RESOURCE_NODE *
305 RsDoDwordMemoryDescriptor (
306     ACPI_PARSE_OBJECT       *Op,
307     UINT32                  CurrentByteOffset)
308 {
309     AML_RESOURCE            *Descriptor;
310     ACPI_PARSE_OBJECT       *InitializerOp;
311     ACPI_PARSE_OBJECT       *MinOp = NULL;
312     ACPI_PARSE_OBJECT       *MaxOp = NULL;
313     ACPI_PARSE_OBJECT       *LengthOp = NULL;
314     ACPI_PARSE_OBJECT       *GranOp = NULL;
315     ASL_RESOURCE_NODE       *Rnode;
316     UINT8                   *OptionalFields;
317     UINT16                  StringLength = 0;
318     UINT32                  OptionIndex = 0;
319     UINT32                  i;
320     BOOLEAN                 ResSourceIndex = FALSE;


323     InitializerOp = Op->Asl.Child;
324     StringLength = RsGetStringDataLength (InitializerOp);
```



```
326     Rnode = RsAllocateResourceNode (
327                 sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);

329     Descriptor = Rnode->Buffer;
330     Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;
331     Descriptor->Address32.ResourceType   = ACPI_ADDRESS_TYPE_MEMORY_RANGE;

333     /*
334      * Initial descriptor length -- may be enlarged if there are
335      * optional fields present
336      */
337     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS32);
338     Descriptor->Address32.ResourceLength = (UINT16)
339         (sizeof (AML_RESOURCE_ADDRESS32) -
340          sizeof (AML_RESOURCE_LARGE_HEADER));


343     /* Process all child initialization nodes */

345     for (i = 0; InitializerOp; i++)
346     {
347         switch (i)
348         {
349         case 0: /* Resource Usage */

351             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
352             break;

354         case 1: /* DecodeType */

356             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 1, 0);
357             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
358                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 1);
359             break;

361         case 2: /* MinType */

363             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
364             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
365                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 2);
366             break;

368         case 3: /* MaxType */

370             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 3, 0);
371             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
372                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 3);
373             break;

375         case 4: /* Memory Type */

377             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
378             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMTYPE,
379                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
380             break;

382         case 5: /* Read/Write Type */

384             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
385             RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
386                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
387             break;

389         case 6: /* Address Granularity */
```

```
391             Descriptor->Address32.Granularity =
392                 (UINT32) InitializerOp->Asl.Value.Integer;
393             RsCreateDwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
394                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Granularity));
395             GranOp = InitializerOp;
396             break;

398         case 7: /* Min Address */

400             Descriptor->Address32.Minimum =
401                 (UINT32) InitializerOp->Asl.Value.Integer;
402             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
403                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Minimum));
404             MinOp = InitializerOp;
405             break;

407         case 8: /* Max Address */

409             Descriptor->Address32.Maximum =
410                 (UINT32) InitializerOp->Asl.Value.Integer;
411             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
412                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Maximum));
413             MaxOp = InitializerOp;
414             break;

416         case 9: /* Translation Offset */

418             Descriptor->Address32.TranslationOffset =
419                 (UINT32) InitializerOp->Asl.Value.Integer;
420             RsCreateDwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
421                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.TranslationOff
422             break;

424         case 10: /* Address Length */

426             Descriptor->Address32.AddressLength =
427                 (UINT32) InitializerOp->Asl.Value.Integer;
428             RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
429                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.AddressLength)
430             LengthOp = InitializerOp;
431             break;

433         case 11: /* ResSourceIndex [Optional Field - BYTE] */

435             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
436             {
437                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
438                 OptionIndex++;
439                 Descriptor->Address32.ResourceLength++;
440                 ResSourceIndex = TRUE;
441             }
442             break;

444         case 12: /* ResSource [Optional Field - STRING] */

446             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
447                 (InitializerOp->Asl.Value.String))
448             {
449                 if (StringLength)
450                 {
451                     Descriptor->Address32.ResourceLength = (UINT16)
452                         (Descriptor->Address32.ResourceLength + StringLength);

454                     strcpy ((char *)
455                         &OptionalFields[OptionIndex],
456                         InitializerOp->Asl.Value.String);
```



```
458                     /* ResourceSourceIndex must also be valid */

460                     if (!ResSourceIndex)
461                     {
462                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
463                             InitializerOp, NULL);
464                     }
465                 }
466             }

468 #if 0
469             /*
470              * Not a valid ResourceSource, ResourceSourceIndex must also
471              * be invalid
472              */
473             else if (ResSourceIndex)
474             {
475                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
476                     InitializerOp, NULL);
477             }
478 #endif
479             break;

481         case 13: /* ResourceTag */

483             UtAttachNamepathToOwner (Op, InitializerOp);
484             break;


487         case 14: /* Address Range */

489             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
490             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMATTRIBUTES,
491                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
492             break;

494         case 15: /* Type */

496             RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
497             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
498                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.SpecificFlags)
499             break;

501         default:

503             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
504             break;
505         }

507         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
508     }

510     /* Validate the Min/Max/Len/Gran values */

512     RsLargeAddressCheck (
513         (UINT64) Descriptor->Address32.Minimum,
514         (UINT64) Descriptor->Address32.Maximum,
515         (UINT64) Descriptor->Address32.AddressLength,
516         (UINT64) Descriptor->Address32.Granularity,
517         Descriptor->Address32.Flags,
518         MinOp, MaxOp, LengthOp, GranOp, Op);

520     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS32) +
521                               OptionIndex + StringLength;
522     return (Rnode);
```

```
}


/*******************************************************************************
 *
 * FUNCTION:    RsDoDwordSpaceDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "DwordSpace" descriptor
 *
 ******************************************************************************/

ASL_RESOURCE_NODE *
RsDoDwordSpaceDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *GranOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT8                   *OptionalFields;
    UINT16                  StringLength = 0;
    UINT32                  OptionIndex = 0;
    UINT32                  i;
    BOOLEAN                 ResSourceIndex = FALSE;


    InitializerOp = Op->Asl.Child;
    StringLength = RsGetStringDataLength (InitializerOp);

    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;

    /*
     * Initial descriptor length -- may be enlarged if there are
     * optional fields present
     */
    OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS32);
    Descriptor->Address32.ResourceLength = (UINT16)
        (sizeof (AML_RESOURCE_ADDRESS32) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Type */

            Descriptor->Address32.ResourceType =
                (UINT8) InitializerOp->Asl.Value.Integer;
            break;
```



```
        case 1: /* Resource Usage */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
            break;

        case 2: /* DecodeType */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 1, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 1);
            break;

        case 3: /* MinType */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 2);
            break;

        case 4: /* MaxType */

            RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 3, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Flags), 3);
            break;

        case 5: /* Type-Specific flags */

            Descriptor->Address32.SpecificFlags =
                (UINT8) InitializerOp->Asl.Value.Integer;
            break;

        case 6: /* Address Granularity */

            Descriptor->Address32.Granularity =
                (UINT32) InitializerOp->Asl.Value.Integer;
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Granularity));
            GranOp = InitializerOp;
            break;

        case 7: /* Min Address */

            Descriptor->Address32.Minimum =
                (UINT32) InitializerOp->Asl.Value.Integer;
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Minimum));
            MinOp = InitializerOp;
            break;

        case 8: /* Max Address */

            Descriptor->Address32.Maximum =
                (UINT32) InitializerOp->Asl.Value.Integer;
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.Maximum));
            MaxOp = InitializerOp;
            break;

        case 9: /* Translation Offset */

            Descriptor->Address32.TranslationOffset =
                (UINT32) InitializerOp->Asl.Value.Integer;
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.TranslationOff
            break;
```

```
656         case 10: /* Address Length */

658             Descriptor->Address32.AddressLength =
659                 (UINT32) InitializerOp->Asl.Value.Integer;
660             RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
661                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address32.AddressLength)
662             LengthOp = InitializerOp;
663             break;

665         case 11: /* ResSourceIndex [Optional Field - BYTE] */

667             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
668             {
669                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
670                 OptionIndex++;
671                 Descriptor->Address32.ResourceLength++;
672                 ResSourceIndex = TRUE;
673             }
674             break;

676         case 12: /* ResSource [Optional Field - STRING] */

678             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
679                 (InitializerOp->Asl.Value.String))
680             {
681                 if (StringLength)
682                 {
683                     Descriptor->Address32.ResourceLength = (UINT16)
684                         (Descriptor->Address32.ResourceLength + StringLength);

686                     strcpy ((char *)
687                         &OptionalFields[OptionIndex],
688                         InitializerOp->Asl.Value.String);

690                     /* ResourceSourceIndex must also be valid */

692                     if (!ResSourceIndex)
693                     {
694                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
695                             InitializerOp, NULL);
696                     }
697                 }
698             }

700 #if 0
701             /*
702              * Not a valid ResourceSource, ResourceSourceIndex must also
703              * be invalid
704              */
705             else if (ResSourceIndex)
706             {
707                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
708                     InitializerOp, NULL);
709             }
710 #endif
711             break;

713         case 13: /* ResourceTag */

715             UtAttachNamepathToOwner (Op, InitializerOp);
716             break;

718         default:

720             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST,
```



```
721                 InitializerOp, NULL);
722             break;
723         }

725         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
726     }

728     /* Validate the Min/Max/Len/Gran values */

730     RsLargeAddressCheck (
731         (UINT64) Descriptor->Address32.Minimum,
732         (UINT64) Descriptor->Address32.Maximum,
733         (UINT64) Descriptor->Address32.AddressLength,
734         (UINT64) Descriptor->Address32.Granularity,
735         Descriptor->Address32.Flags,
736         MinOp, MaxOp, LengthOp, GranOp, Op);

738     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS32) +
739                           OptionIndex + StringLength;
740     return (Rnode);
741 }
```

```
**********************************************************
   21325 Thu Dec 26 13:48:33 2013
new/usr/src/common/acpica/compiler/aslrestype2e.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2e - Large Extended address resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2e")

/*
 * This module contains the Extended (64-bit) address space descriptors:
 *
 * ExtendedIO
 * ExtendedMemory
 * ExtendedSpace
 */

/******************************************************************************
 *
 * FUNCTION:    RsDoExtendedIoDescriptor
```



```
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "ExtendedIO" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoExtendedIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *GranOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT16                  StringLength = 0;
    UINT32                  i;


    InitializerOp = Op->Asl.Child;
    StringLength = RsGetStringDataLength (InitializerOp);

    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->ExtAddress64.DescriptorType  = ACPI_RESOURCE_NAME_EXTENDED_ADDRE
    Descriptor->ExtAddress64.ResourceType    = ACPI_ADDRESS_TYPE_IO_RANGE;
    Descriptor->ExtAddress64.RevisionID      = AML_RESOURCE_EXTENDED_ADDRESS_REV

    Descriptor->ExtAddress64.ResourceLength  = (UINT16)
        (sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Usage */

            RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 0, 1)
            break;

        case 1: /* MinType */

            RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 2, 0)
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 2);
            break;

        case 2: /* MaxType */

            RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 3, 0)
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 3);
            break;
```

```
128         case 3: /* DecodeType */
130             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 1, 0)
131             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
132                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 1);
133             break;
135         case 4: /* Range Type */
137             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
138             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
139                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
140             break;
142         case 5: /* Address Granularity */
144             Descriptor->ExtAddress64.Granularity = InitializerOp->Asl.Value.Inte
145             RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
146                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Granularity
147             GranOp = InitializerOp;
148            break;
150         case 6: /* Address Min */
152             Descriptor->ExtAddress64.Minimum = InitializerOp->Asl.Value.Integer;
153             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
154                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Minimum));
155             MinOp = InitializerOp;
156             break;
158         case 7: /* Address Max */
160             Descriptor->ExtAddress64.Maximum = InitializerOp->Asl.Value.Integer;
161             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
162                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Maximum));
163             MaxOp = InitializerOp;
164             break;
166         case 8: /* Translation Offset */
168             Descriptor->ExtAddress64.TranslationOffset = InitializerOp->Asl.Valu
169             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
170                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Translation
171             break;
173         case 9: /* Address Length */
175             Descriptor->ExtAddress64.AddressLength = InitializerOp->Asl.Value.In
176             RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
177                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.AddressLeng
178             LengthOp = InitializerOp;
179             break;
181         case 10: /* Type-Specific Attributes */
183             Descriptor->ExtAddress64.TypeSpecific = InitializerOp->Asl.Value.Int
184             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TYPESPECIFICATTRIBUTE
185                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.TypeSpecifi
186             break;
188         case 11: /* ResourceTag */
190             UtAttachNamepathToOwner (Op, InitializerOp);
191             break;
```



```
193         case 12: /* Type */
195             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
196             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
197                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
198             break;
200         case 13: /* Translation Type */
202             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
203             RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
204                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
205             break;
207         default:
209             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
210             break;
211         }
213         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
214     }
216     /* Validate the Min/Max/Len/Gran values */
218     RsLargeAddressCheck (
219         Descriptor->ExtAddress64.Minimum,
220         Descriptor->ExtAddress64.Maximum,
221         Descriptor->ExtAddress64.AddressLength,
222         Descriptor->ExtAddress64.Granularity,
223         Descriptor->ExtAddress64.Flags,
224         MinOp, MaxOp, LengthOp, GranOp, Op);
226     Rnode->BufferLength = sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + StringLengt
227     return (Rnode);
228 }
231 /*******************************************************************************
232  *
233  * FUNCTION:    RsDoExtendedMemoryDescriptor
234  *
235  * PARAMETERS:  Op                  - Parent resource descriptor parse node
236  *              CurrentByteOffset   - Offset into the resource template AML
237  *                                    buffer (to track references to the desc)
238  *
239  * RETURN:      Completed resource node
240  *
241  * DESCRIPTION: Construct a long "ExtendedMemory" descriptor
242  *
243  ******************************************************************************/
245 ASL_RESOURCE_NODE *
246 RsDoExtendedMemoryDescriptor (
247     ACPI_PARSE_OBJECT       *Op,
248     UINT32                  CurrentByteOffset)
249 {
250     AML_RESOURCE            *Descriptor;
251     ACPI_PARSE_OBJECT       *InitializerOp;
252     ACPI_PARSE_OBJECT       *MinOp = NULL;
253     ACPI_PARSE_OBJECT       *MaxOp = NULL;
254     ACPI_PARSE_OBJECT       *LengthOp = NULL;
255     ACPI_PARSE_OBJECT       *GranOp = NULL;
256     ASL_RESOURCE_NODE       *Rnode;
257     UINT16                  StringLength = 0;
258     UINT32                  i;
```

```
261     InitializerOp = Op->Asl.Child;
262     StringLength = RsGetStringDataLength (InitializerOp);
263
264     Rnode = RsAllocateResourceNode (
265                 sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + 1 + StringLength);
266
267     Descriptor = Rnode->Buffer;
268     Descriptor->ExtAddress64.DescriptorType  = ACPI_RESOURCE_NAME_EXTENDED_ADDRE
269     Descriptor->ExtAddress64.ResourceType    = ACPI_ADDRESS_TYPE_MEMORY_RANGE;
270     Descriptor->ExtAddress64.RevisionID      = AML_RESOURCE_EXTENDED_ADDRESS_REV
271
272     Descriptor->ExtAddress64.ResourceLength  = (UINT16)
273         (sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) -
274          sizeof (AML_RESOURCE_LARGE_HEADER));
275
276     /* Process all child initialization nodes */
277
278     for (i = 0; InitializerOp; i++)
279     {
280         switch (i)
281         {
282         case 0: /* Resource Usage */
283
284             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 0, 1)
285             break;
286
287         case 1: /* DecodeType */
288
289             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 1, 0)
290             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
291                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 1);
292             break;
293
294         case 2: /* MinType */
295
296             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 2, 0)
297             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
298                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 2);
299             break;
300
301         case 3: /* MaxType */
302
303             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 3, 0)
304             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
305                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 3);
306             break;
307
308         case 4: /* Memory Type */
309
310             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
311             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMTYPE,
312                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
313             break;
314
315         case 5: /* Read/Write Type */
316
317             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
318             RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
319                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
320             break;
321
322         case 6: /* Address Granularity */
323
324             Descriptor->ExtAddress64.Granularity = InitializerOp->Asl.Value.Inte
```



```
325             RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
326                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Granularity
327             GranOp = InitializerOp;
328             break;
329
330         case 7: /* Min Address */
331
332             Descriptor->ExtAddress64.Minimum = InitializerOp->Asl.Value.Integer;
333             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
334                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Minimum));
335             MinOp = InitializerOp;
336             break;
337
338         case 8: /* Max Address */
339
340             Descriptor->ExtAddress64.Maximum = InitializerOp->Asl.Value.Integer;
341             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
342                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Maximum));
343             MaxOp = InitializerOp;
344             break;
345
346         case 9: /* Translation Offset */
347
348             Descriptor->ExtAddress64.TranslationOffset = InitializerOp->Asl.Valu
349             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
350                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Translation
351             break;
352
353         case 10: /* Address Length */
354
355             Descriptor->ExtAddress64.AddressLength = InitializerOp->Asl.Value.In
356             RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
357                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.AddressLeng
358             LengthOp = InitializerOp;
359             break;
360
361         case 11: /* Type-Specific Attributes */
362
363             Descriptor->ExtAddress64.TypeSpecific = InitializerOp->Asl.Value.Int
364             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TYPESPECIFICATTRIBUTE
365                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.TypeSpecifi
366             break;
367
368         case 12: /* ResourceTag */
369
370             UtAttachNamepathToOwner (Op, InitializerOp);
371             break;
372
373
374         case 13: /* Address Range */
375
376             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
377             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMATTRIBUTES,
378                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
379             break;
380
381         case 14: /* Type */
382
383             RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerO
384             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
385                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.SpecificFla
386             break;
387
388         default:
389
390             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
```

```
391             break;
392         }
394         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
395     }
397     /* Validate the Min/Max/Len/Gran values */
399     RsLargeAddressCheck (
400         Descriptor->ExtAddress64.Minimum,
401         Descriptor->ExtAddress64.Maximum,
402         Descriptor->ExtAddress64.AddressLength,
403         Descriptor->ExtAddress64.Granularity,
404         Descriptor->ExtAddress64.Flags,
405         MinOp, MaxOp, LengthOp, GranOp, Op);
407     Rnode->BufferLength = sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + StringLengt
408     return (Rnode);
409 }
412 /*******************************************************************************
413  *
414  * FUNCTION:    RsDoExtendedSpaceDescriptor
415  *
416  * PARAMETERS:  Op                  - Parent resource descriptor parse node
417  *              CurrentByteOffset   - Offset into the resource template AML
418  *                                    buffer (to track references to the desc)
419  *
420  * RETURN:      Completed resource node
421  *
422  * DESCRIPTION: Construct a long "ExtendedSpace" descriptor
423  *
424  ******************************************************************************/
426 ASL_RESOURCE_NODE *
427 RsDoExtendedSpaceDescriptor (
428     ACPI_PARSE_OBJECT       *Op,
429     UINT32                  CurrentByteOffset)
430 {
431     AML_RESOURCE            *Descriptor;
432     ACPI_PARSE_OBJECT       *InitializerOp;
433     ACPI_PARSE_OBJECT       *MinOp = NULL;
434     ACPI_PARSE_OBJECT       *MaxOp = NULL;
435     ACPI_PARSE_OBJECT       *LengthOp = NULL;
436     ACPI_PARSE_OBJECT       *GranOp = NULL;
437     ASL_RESOURCE_NODE       *Rnode;
438     UINT16                  StringLength = 0;
439     UINT32                  i;
442     InitializerOp = Op->Asl.Child;
443     StringLength = RsGetStringDataLength (InitializerOp);
445     Rnode = RsAllocateResourceNode (
446                 sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + 1 + StringLength);
448     Descriptor = Rnode->Buffer;
449     Descriptor->ExtAddress64.DescriptorType  = ACPI_RESOURCE_NAME_EXTENDED_ADDRE
450     Descriptor->ExtAddress64.RevisionID      = AML_RESOURCE_EXTENDED_ADDRESS_REV
452     Descriptor->ExtAddress64.ResourceLength  = (UINT16)
453         (sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) -
454          sizeof (AML_RESOURCE_LARGE_HEADER));
456     /* Process all child initialization nodes */
```



```
458     for (i = 0; InitializerOp; i++)
459     {
460         switch (i)
461         {
462         case 0: /* Resource Type */
464             Descriptor->ExtAddress64.ResourceType =
465                 (UINT8) InitializerOp->Asl.Value.Integer;
466             break;
468         case 1: /* Resource Usage */
470             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 0, 1)
471             break;
473         case 2: /* DecodeType */
475             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 1, 0)
476             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
477                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 1);
478             break;
480         case 3: /* MinType */
482             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 2, 0)
483             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
484                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 2);
485             break;
487         case 4: /* MaxType */
489             RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 3, 0)
490             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
491                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Flags), 3);
492             break;
494         case 5: /* Type-Specific flags */
496             Descriptor->ExtAddress64.SpecificFlags =
497                 (UINT8) InitializerOp->Asl.Value.Integer;
498             break;
500         case 6: /* Address Granularity */
502             Descriptor->ExtAddress64.Granularity = InitializerOp->Asl.Value.Inte
503             RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
504                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Granularity
505             GranOp = InitializerOp;
506             break;
508         case 7: /* Min Address */
510             Descriptor->ExtAddress64.Minimum = InitializerOp->Asl.Value.Integer;
511             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
512                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Minimum));
513             MinOp = InitializerOp;
514             break;
516         case 8: /* Max Address */
518             Descriptor->ExtAddress64.Maximum = InitializerOp->Asl.Value.Integer;
519             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
520                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Maximum));
521             MaxOp = InitializerOp;
522             break;
```

```
524         case 9: /* Translation Offset */

526             Descriptor->ExtAddress64.TranslationOffset = InitializerOp->Asl.Valu
527             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
528                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.Translation
529             break;

531         case 10: /* Address Length */

533             Descriptor->ExtAddress64.AddressLength = InitializerOp->Asl.Value.In
534             RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
535                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.AddressLeng
536             LengthOp = InitializerOp;
537             break;

539         case 11: /* Type-Specific Attributes */

541             Descriptor->ExtAddress64.TypeSpecific = InitializerOp->Asl.Value.Int
542             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TYPESPECIFICATTRIBUTE
543                 CurrentByteOffset + ASL_RESDESC_OFFSET (ExtAddress64.TypeSpecifi
544             break;

546         case 12: /* ResourceTag */

548             UtAttachNamepathToOwner (Op, InitializerOp);
549             break;

551         default:

553             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
554             break;
555         }

557         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
558     }

560     /* Validate the Min/Max/Len/Gran values */

562     RsLargeAddressCheck (
563         Descriptor->ExtAddress64.Minimum,
564         Descriptor->ExtAddress64.Maximum,
565         Descriptor->ExtAddress64.AddressLength,
566         Descriptor->ExtAddress64.Granularity,
567         Descriptor->ExtAddress64.Flags,
568         MinOp, MaxOp, LengthOp, GranOp, Op);

570     Rnode->BufferLength = sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + StringLengt
571     return (Rnode);
572 }
```

```
**********************************************************
   25533 Thu Dec 26 13:48:33 2013
new/usr/src/common/acpica/compiler/aslrestype2q.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2q - Large QWord address resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2q")

/*
 * This module contains the QWord (64-bit) address space descriptors:
 *
 * QWordIO
 * QWordMemory
 * QWordSpace
 */

/******************************************************************************
 *
```



```
 * FUNCTION:    RsDoQwordIoDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "QwordIO" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoQwordIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *GranOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT8                   *OptionalFields;
    UINT16                  StringLength = 0;
    UINT32                  OptionIndex = 0;
    UINT32                  i;
    BOOLEAN                 ResSourceIndex = FALSE;


    InitializerOp = Op->Asl.Child;
    StringLength = RsGetStringDataLength (InitializerOp);

    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_ADDRESS64) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->Address64.DescriptorType  = ACPI_RESOURCE_NAME_ADDRESS64;
    Descriptor->Address64.ResourceType    = ACPI_ADDRESS_TYPE_IO_RANGE;

    /*
     * Initial descriptor length -- may be enlarged if there are
     * optional fields present
     */
    OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS64);
    Descriptor->Address64.ResourceLength = (UINT16)
        (sizeof (AML_RESOURCE_ADDRESS64) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Usage */

            RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 0, 1);
            break;

        case 1: /* MinType */

            RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 2);
```

```
            break;

        case 2: /* MaxType */

            RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 3, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 3);
            break;

        case 3: /* DecodeType */

            RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 1, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 1);
            break;

        case 4: /* Range Type */

            RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
            break;

        case 5: /* Address Granularity */

            Descriptor->Address64.Granularity = InitializerOp->Asl.Value.Integer
            RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Granularity));
            GranOp = InitializerOp;
            break;

        case 6: /* Address Min */

            Descriptor->Address64.Minimum = InitializerOp->Asl.Value.Integer;
            RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Minimum));
            MinOp = InitializerOp;
            break;

        case 7: /* Address Max */

            Descriptor->Address64.Maximum = InitializerOp->Asl.Value.Integer;
            RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Maximum));
            MaxOp = InitializerOp;
            break;

        case 8: /* Translation Offset */

            Descriptor->Address64.TranslationOffset = InitializerOp->Asl.Value.I
            RsCreateByteField (InitializerOp, ACPI_RESTAG_TRANSLATION,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.TranslationOff
            break;

        case 9: /* Address Length */

            Descriptor->Address64.AddressLength = InitializerOp->Asl.Value.Integ
            RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.AddressLength)
            LengthOp = InitializerOp;
            break;

        case 10: /* ResSourceIndex [Optional Field - BYTE] */

            if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
            {
```



```
                OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
                OptionIndex++;
                Descriptor->Address64.ResourceLength++;
                ResSourceIndex = TRUE;
            }
            break;

        case 11: /* ResSource [Optional Field - STRING] */

            if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
                (InitializerOp->Asl.Value.String))
            {
                if (StringLength)
                {
                    Descriptor->Address64.ResourceLength = (UINT16)
                        (Descriptor->Address64.ResourceLength + StringLength);

                    strcpy ((char *)
                        &OptionalFields[OptionIndex],
                        InitializerOp->Asl.Value.String);

                    /* ResourceSourceIndex must also be valid */

                    if (!ResSourceIndex)
                    {
                        AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
                            InitializerOp, NULL);
                    }
                }
            }

#if 0
            /*
             * Not a valid ResourceSource, ResourceSourceIndex must also
             * be invalid
             */
            else if (ResSourceIndex)
            {
                AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
                    InitializerOp, NULL);
            }
#endif
            break;

        case 12: /* ResourceTag */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;

        case 13: /* Type */

            RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
            RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
            break;

        case 14: /* Translation Type */

            RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
            RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
            break;

        default:

            AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
```

```
259             break;
260         }
262         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
263     }
265     /* Validate the Min/Max/Len/Gran values */
267     RsLargeAddressCheck (
268         Descriptor->Address64.Minimum,
269         Descriptor->Address64.Maximum,
270         Descriptor->Address64.AddressLength,
271         Descriptor->Address64.Granularity,
272         Descriptor->Address64.Flags,
273         MinOp, MaxOp, LengthOp, GranOp, Op);
275     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS64) +
276                           OptionIndex + StringLength;
277     return (Rnode);
278 }
281 /*******************************************************************************
282  *
283  * FUNCTION:    RsDoQwordMemoryDescriptor
284  *
285  * PARAMETERS:  Op                  - Parent resource descriptor parse node
286  *              CurrentByteOffset   - Offset into the resource template AML
287  *                                    buffer (to track references to the desc)
288  *
289  * RETURN:      Completed resource node
290  *
291  * DESCRIPTION: Construct a long "QwordMemory" descriptor
292  *
293  ******************************************************************************/
295 ASL_RESOURCE_NODE *
296 RsDoQwordMemoryDescriptor (
297     ACPI_PARSE_OBJECT       *Op,
298     UINT32                  CurrentByteOffset)
299 {
300     AML_RESOURCE            *Descriptor;
301     ACPI_PARSE_OBJECT       *InitializerOp;
302     ACPI_PARSE_OBJECT       *MinOp = NULL;
303     ACPI_PARSE_OBJECT       *MaxOp = NULL;
304     ACPI_PARSE_OBJECT       *LengthOp = NULL;
305     ACPI_PARSE_OBJECT       *GranOp = NULL;
306     ASL_RESOURCE_NODE       *Rnode;
307     UINT8                   *OptionalFields;
308     UINT16                  StringLength = 0;
309     UINT32                  OptionIndex = 0;
310     UINT32                  i;
311     BOOLEAN                 ResSourceIndex = FALSE;
314     InitializerOp = Op->Asl.Child;
315     StringLength = RsGetStringDataLength (InitializerOp);
317     Rnode = RsAllocateResourceNode (
318                 sizeof (AML_RESOURCE_ADDRESS64) + 1 + StringLength);
320     Descriptor = Rnode->Buffer;
321     Descriptor->Address64.DescriptorType  = ACPI_RESOURCE_NAME_ADDRESS64;
322     Descriptor->Address64.ResourceType    = ACPI_ADDRESS_TYPE_MEMORY_RANGE;
324     /*
```



```
325      * Initial descriptor length -- may be enlarged if there are
326      * optional fields present
327      */
328     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS64);
329     Descriptor->Address64.ResourceLength = (UINT16)
330         (sizeof (AML_RESOURCE_ADDRESS64) -
331          sizeof (AML_RESOURCE_LARGE_HEADER));
333     /* Process all child initialization nodes */
335     for (i = 0; InitializerOp; i++)
336     {
337         switch (i)
338         {
339         case 0: /* Resource Usage */
341             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 0, 1);
342             break;
344         case 1: /* DecodeType */
346             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 1, 0);
347             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
348                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 1);
349             break;
351         case 2: /* MinType */
353             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 2, 0);
354             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
355                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 2);
356             break;
358         case 3: /* MaxType */
360             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 3, 0);
361             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
362                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 3);
363             break;
365         case 4: /* Memory Type */
367             RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
368             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMTYPE,
369                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
370             break;
372         case 5: /* Read/Write Type */
374             RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
375             RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
376                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
377             break;
379         case 6: /* Address Granularity */
381             Descriptor->Address64.Granularity = InitializerOp->Asl.Value.Integer
382             RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
383                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Granularity));
384             GranOp = InitializerOp;
385             break;
387         case 7: /* Min Address */
389             Descriptor->Address64.Minimum = InitializerOp->Asl.Value.Integer;
390             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
```

```
391                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Minimum));
392             MinOp = InitializerOp;
393             break;

395         case 8: /* Max Address */

397             Descriptor->Address64.Maximum = InitializerOp->Asl.Value.Integer;
398             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
399                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Maximum));
400             MaxOp = InitializerOp;
401             break;

403         case 9: /* Translation Offset */

405             Descriptor->Address64.TranslationOffset = InitializerOp->Asl.Value.I
406             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
407                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.TranslationOff
408             break;

410         case 10: /* Address Length */

412             Descriptor->Address64.AddressLength = InitializerOp->Asl.Value.Integ
413             RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
414                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.AddressLength)
415             LengthOp = InitializerOp;
416             break;

418         case 11: /* ResSourceIndex [Optional Field - BYTE] */

420             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
421             {
422                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
423                 OptionIndex++;
424                 Descriptor->Address64.ResourceLength++;
425                 ResSourceIndex = TRUE;
426             }
427             break;

429         case 12: /* ResSource [Optional Field - STRING] */

431             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
432                 (InitializerOp->Asl.Value.String))
433             {
434                 if (StringLength)
435                 {
436                     Descriptor->Address64.ResourceLength = (UINT16)
437                         (Descriptor->Address64.ResourceLength + StringLength);

439                     strcpy ((char *)
440                         &OptionalFields[OptionIndex],
441                         InitializerOp->Asl.Value.String);

443                     /* ResourceSourceIndex must also be valid */

445                     if (!ResSourceIndex)
446                     {
447                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
448                             InitializerOp, NULL);
449                     }
450                 }
451             }

453 #if 0
454             /*
455              * Not a valid ResourceSource, ResourceSourceIndex must also
456              * be invalid
```



```
457              */
458             else if (ResSourceIndex)
459             {
460                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
461                     InitializerOp, NULL);
462             }
463 #endif
464             break;

466         case 13: /* ResourceTag */

468             UtAttachNamepathToOwner (Op, InitializerOp);
469             break;


472         case 14: /* Address Range */

474             RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
475             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMATTRIBUTES,
476                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
477             break;

479         case 15: /* Type */

481             RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
482             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
483                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.SpecificFlags)
484             break;

486         default:

488             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
489             break;
490         }

492         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
493     }

495     /* Validate the Min/Max/Len/Gran values */

497     RsLargeAddressCheck (
498         Descriptor->Address64.Minimum,
499         Descriptor->Address64.Maximum,
500         Descriptor->Address64.AddressLength,
501         Descriptor->Address64.Granularity,
502         Descriptor->Address64.Flags,
503         MinOp, MaxOp, LengthOp, GranOp, Op);

505     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS64) +
506                               OptionIndex + StringLength;
507     return (Rnode);
508 }


511 /*******************************************************************************
512  *
513  * FUNCTION:    RsDoQwordSpaceDescriptor
514  *
515  * PARAMETERS:  Op                  - Parent resource descriptor parse node
516  *              CurrentByteOffset   - Offset into the resource template AML
517  *                                    buffer (to track references to the desc)
518  *
519  * RETURN:      Completed resource node
520  *
521  * DESCRIPTION: Construct a long "QwordSpace" descriptor
522  *
```

```
523  *****************************************************************************/

525 ASL_RESOURCE_NODE *
526 RsDoQwordSpaceDescriptor (
527     ACPI_PARSE_OBJECT       *Op,
528     UINT32                  CurrentByteOffset)
529 {
530     AML_RESOURCE            *Descriptor;
531     ACPI_PARSE_OBJECT       *InitializerOp;
532     ACPI_PARSE_OBJECT       *MinOp = NULL;
533     ACPI_PARSE_OBJECT       *MaxOp = NULL;
534     ACPI_PARSE_OBJECT       *LengthOp = NULL;
535     ACPI_PARSE_OBJECT       *GranOp = NULL;
536     ASL_RESOURCE_NODE       *Rnode;
537     UINT8                   *OptionalFields;
538     UINT16                  StringLength = 0;
539     UINT32                  OptionIndex = 0;
540     UINT32                  i;
541     BOOLEAN                 ResSourceIndex = FALSE;


544     InitializerOp = Op->Asl.Child;
545     StringLength = RsGetStringDataLength (InitializerOp);

547     Rnode = RsAllocateResourceNode (
548                 sizeof (AML_RESOURCE_ADDRESS64) + 1 + StringLength);

550     Descriptor = Rnode->Buffer;
551     Descriptor->Address64.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS64;

553     /*
554      * Initial descriptor length -- may be enlarged if there are
555      * optional fields present
556      */
557     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS64);
558     Descriptor->Address64.ResourceLength = (UINT16)
559         (sizeof (AML_RESOURCE_ADDRESS64) -
560          sizeof (AML_RESOURCE_LARGE_HEADER));

562     /* Process all child initialization nodes */

564     for (i = 0; InitializerOp; i++)
565     {
566         switch (i)
567         {
568         case 0: /* Resource Type */

570             Descriptor->Address64.ResourceType =
571                 (UINT8) InitializerOp->Asl.Value.Integer;
572             break;

574         case 1: /* Resource Usage */

576             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 0, 1);
577             break;

579         case 2: /* DecodeType */

581             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 1, 0);
582             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
583                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 1);
584             break;

586         case 3: /* MinType */

588             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 2, 0);
```



```
589             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
590                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 2);
591             break;

593         case 4: /* MaxType */

595             RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 3, 0);
596             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
597                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Flags), 3);
598             break;

600         case 5: /* Type-Specific flags */

602             Descriptor->Address64.SpecificFlags =
603                 (UINT8) InitializerOp->Asl.Value.Integer;
604             break;

606         case 6: /* Address Granularity */

608             Descriptor->Address64.Granularity = InitializerOp->Asl.Value.Integer
609             RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
610                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Granularity));
611             GranOp = InitializerOp;
612             break;

614         case 7: /* Min Address */

616             Descriptor->Address64.Minimum = InitializerOp->Asl.Value.Integer;
617             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
618                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Minimum));
619             MinOp = InitializerOp;
620             break;

622         case 8: /* Max Address */

624             Descriptor->Address64.Maximum = InitializerOp->Asl.Value.Integer;
625             RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
626                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.Maximum));
627             MaxOp = InitializerOp;
628             break;

630         case 9: /* Translation Offset */

632             Descriptor->Address64.TranslationOffset = InitializerOp->Asl.Value.I
633             RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
634                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.TranslationOff
635             break;

637         case 10: /* Address Length */

639             Descriptor->Address64.AddressLength = InitializerOp->Asl.Value.Integ
640             RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
641                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address64.AddressLength)
642             LengthOp = InitializerOp;
643             break;

645         case 11: /* ResSourceIndex [Optional Field - BYTE] */

647             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
648             {
649                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
650                 OptionIndex++;
651                 Descriptor->Address64.ResourceLength++;
652                 ResSourceIndex = TRUE;
653             }
654             break;
```

```
656         case 12: /* ResSource [Optional Field - STRING] */

658             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
659                 (InitializerOp->Asl.Value.String))
660             {
661                 if (StringLength)
662                 {
663                     Descriptor->Address64.ResourceLength = (UINT16)
664                         (Descriptor->Address64.ResourceLength + StringLength);

666                     strcpy ((char *)
667                         &OptionalFields[OptionIndex],
668                         InitializerOp->Asl.Value.String);

670                     /* ResourceSourceIndex must also be valid */

672                     if (!ResSourceIndex)
673                     {
674                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
675                             InitializerOp, NULL);
676                     }
677                 }
678             }

680 #if 0
681             /*
682              * Not a valid ResourceSource, ResourceSourceIndex must also
683              * be invalid
684              */
685             else if (ResSourceIndex)
686             {
687                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
688                     InitializerOp, NULL);
689             }
690 #endif
691             break;

693         case 13: /* ResourceTag */

695             UtAttachNamepathToOwner (Op, InitializerOp);
696             break;

698         default:

700             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
701             break;
702         }

704         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
705     }

707     /* Validate the Min/Max/Len/Gran values */

709     RsLargeAddressCheck (
710         Descriptor->Address64.Minimum,
711         Descriptor->Address64.Maximum,
712         Descriptor->Address64.AddressLength,
713         Descriptor->Address64.Granularity,
714         Descriptor->Address64.Flags,
715         MinOp, MaxOp, LengthOp, GranOp, Op);

717     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS64) +
718                           OptionIndex + StringLength;
719     return (Rnode);
720 }
```

```
**********************************************************
   42048 Thu Dec 26 13:48:34 2013
new/usr/src/common/acpica/compiler/aslrestype2s.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2s - Serial Large resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2s")


static UINT16
RsGetBufferDataLength (
    ACPI_PARSE_OBJECT       *InitializerOp);

static UINT16
RsGetInterruptDataLength (
    ACPI_PARSE_OBJECT       *InitializerOp);

static BOOLEAN
```



```
RsGetVendorData (
    ACPI_PARSE_OBJECT       *InitializerOp,
    UINT8                   *VendorData,
    ACPI_SIZE               DescriptorOffset);

/*
 * This module contains descriptors for serial buses and GPIO:
 *
 * GpioInt
 * GpioIo
 * I2cSerialBus
 * SpiSerialBus
 * UartSerialBus
 */


/******************************************************************************
 *
 * FUNCTION:    RsGetBufferDataLength
 *
 * PARAMETERS:  InitializerOp       - Current parse op, start of the resource
 *                                    descriptor
 *
 * RETURN:      Length of the data buffer
 *
 * DESCRIPTION: Get the length of a RawDataBuffer, used for vendor data.
 *
 *****************************************************************************/

static UINT16
RsGetBufferDataLength (
    ACPI_PARSE_OBJECT       *InitializerOp)
{
    UINT16                  ExtraDataSize = 0;
    ACPI_PARSE_OBJECT       *DataList;


    /* Find the byte-initializer list */

    while (InitializerOp)
    {
        if (InitializerOp->Asl.ParseOpcode == PARSEOP_DATABUFFER)
        {
            /* First child is the optional length (ignore it here) */

            DataList = InitializerOp->Asl.Child;
            DataList = ASL_GET_PEER_NODE (DataList);

            /* Count the data items (each one is a byte of data) */

            while (DataList)
            {
                ExtraDataSize++;
                DataList = ASL_GET_PEER_NODE (DataList);
            }

            return (ExtraDataSize);
        }

        InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
    }

    return (ExtraDataSize);
}
```

```
128 /*******************************************************************************
129  *
130  * FUNCTION:    RsGetInterruptDataLength
131  *
132  * PARAMETERS:  InitializerOp       - Current parse op, start of the resource
133  *                                    descriptor
134  *
135  * RETURN:      Length of the interrupt data list
136  *
137  * DESCRIPTION: Get the length of a list of interrupt DWORDs for the GPIO
138  *              descriptors.
139  *
140  ******************************************************************************/

142 static UINT16
143 RsGetInterruptDataLength (
144     ACPI_PARSE_OBJECT       *InitializerOp)
145 {
146     UINT16                  InterruptLength;
147     UINT32                  i;


150     /* Count the interrupt numbers */

152     InterruptLength = 0;
153     for (i = 0; InitializerOp; i++)
154     {
155         InitializerOp = ASL_GET_PEER_NODE (InitializerOp);

157         /* Interrupt list starts at offset 10 (Gpio descriptors) */

159         if (i >= 10)
160         {
161             InterruptLength += 2;
162         }
163     }

165     return (InterruptLength);
166 }


169 /*******************************************************************************
170  *
171  * FUNCTION:    RsGetVendorData
172  *
173  * PARAMETERS:  InitializerOp       - Current parse op, start of the resource
174  *                                    descriptor.
175  *              VendorData          - Where the vendor data is returned
176  *              DescriptorOffset    - Where vendor data begins in descriptor
177  *
178  * RETURN:      TRUE if valid vendor data was returned, FALSE otherwise.
179  *
180  * DESCRIPTION: Extract the vendor data and construct a vendor data buffer.
181  *
182  ******************************************************************************/

184 static BOOLEAN
185 RsGetVendorData (
186     ACPI_PARSE_OBJECT       *InitializerOp,
187     UINT8                   *VendorData,
188     ACPI_SIZE               DescriptorOffset)
189 {
190     ACPI_PARSE_OBJECT       *BufferOp;
191     UINT32                  SpecifiedLength = ACPI_UINT32_MAX;
192     UINT16                  ActualLength = 0;
```



```
195     /* Vendor Data field is always optional */

197     if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
198     {
199         return (FALSE);
200     }

202     BufferOp = InitializerOp->Asl.Child;
203     if (!BufferOp)
204     {
205         AslError (ASL_ERROR, ASL_MSG_SYNTAX, InitializerOp, "");
206         return (FALSE);
207     }

209     /* First child is the optional buffer length (WORD) */

211     if (BufferOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
212     {
213         SpecifiedLength = (UINT16) BufferOp->Asl.Value.Integer;
214     }

216     /* Insert field tag _VEN */

218     RsCreateByteField (InitializerOp, ACPI_RESTAG_VENDORDATA,
219         (UINT16) DescriptorOffset);

221     /* Walk the list of buffer initializers (each is one byte) */

223     BufferOp = RsCompleteNodeAndGetNext (BufferOp);
224     if (BufferOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
225     {
226         while (BufferOp)
227         {
228             *VendorData = (UINT8) BufferOp->Asl.Value.Integer;
229             VendorData++;
230             ActualLength++;
231             BufferOp = RsCompleteNodeAndGetNext (BufferOp);
232         }
233     }

235     /* Length validation. Buffer cannot be of zero length */

237     if ((SpecifiedLength == 0) ||
238         ((SpecifiedLength == ACPI_UINT32_MAX) && (ActualLength == 0)))
239     {
240         AslError (ASL_ERROR, ASL_MSG_BUFFER_LENGTH, InitializerOp, NULL);
241         return (FALSE);
242     }

244     if (SpecifiedLength != ACPI_UINT32_MAX)
245     {
246         /* ActualLength > SpecifiedLength -> error */

248         if (ActualLength > SpecifiedLength)
249         {
250             AslError (ASL_ERROR, ASL_MSG_LIST_LENGTH_LONG, InitializerOp, NULL);
251             return (FALSE);
252         }

254         /* ActualLength < SpecifiedLength -> remark */

256         else if (ActualLength < SpecifiedLength)
257         {
258             AslError (ASL_REMARK, ASL_MSG_LIST_LENGTH_SHORT, InitializerOp, NULL
259             return (FALSE);
```

```
260         }
261     }

263     return (TRUE);
264 }


267 /*****************************************************************************
268  *
269  * FUNCTION:    RsDoGpioIntDescriptor
270  *
271  * PARAMETERS:  Op                  - Parent resource descriptor parse node
272  *              CurrentByteOffset   - Offset into the resource template AML
273  *                                    buffer (to track references to the desc)
274  *
275  * RETURN:      Completed resource node
276  *
277  * DESCRIPTION: Construct a long "GpioInt" descriptor
278  *
279  ****************************************************************************/

281 ASL_RESOURCE_NODE *
282 RsDoGpioIntDescriptor (
283     ACPI_PARSE_OBJECT       *Op,
284     UINT32                  CurrentByteOffset)
285 {
286     AML_RESOURCE            *Descriptor;
287     ACPI_PARSE_OBJECT       *InitializerOp;
288     ASL_RESOURCE_NODE       *Rnode;
289     char                    *ResourceSource = NULL;
290     UINT8                   *VendorData = NULL;
291     UINT16                  *InterruptList = NULL;
292     UINT16                  ResSourceLength;
293     UINT16                  VendorLength;
294     UINT16                  InterruptLength;
295     UINT16                  DescriptorSize;
296     UINT32                  i;


299     InitializerOp = Op->Asl.Child;

301     /*
302      * Calculate lengths for fields that have variable length:
303      * 1) Resource Source string
304      * 2) Vendor Data buffer
305      * 3) PIN (interrupt) list
306      */
307     ResSourceLength = RsGetStringDataLength (InitializerOp);
308     VendorLength = RsGetBufferDataLength (InitializerOp);
309     InterruptLength = RsGetInterruptDataLength (InitializerOp);

311     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_GPIO) +
312         ResSourceLength + VendorLength + InterruptLength;

314     /* Allocate the local resource node and initialize */

316     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_

318     Descriptor = Rnode->Buffer;
319     Descriptor->Gpio.ResourceLength  = DescriptorSize;
320     Descriptor->Gpio.DescriptorType  = ACPI_RESOURCE_NAME_GPIO;
321     Descriptor->Gpio.RevisionId      = AML_RESOURCE_GPIO_REVISION;
322     Descriptor->Gpio.ConnectionType  = AML_RESOURCE_GPIO_TYPE_INT;

324     /* Build pointers to optional areas */
```



```
326     InterruptList = ACPI_ADD_PTR (UINT16, Descriptor, sizeof (AML_RESOURCE_GPIO)
327     ResourceSource = ACPI_ADD_PTR (char, InterruptList, InterruptLength);
328     VendorData = ACPI_ADD_PTR (UINT8, ResourceSource, ResSourceLength);

330     /* Setup offsets within the descriptor */

332     Descriptor->Gpio.PinTableOffset = (UINT16)
333         ACPI_PTR_DIFF (InterruptList, Descriptor);

335     Descriptor->Gpio.ResSourceOffset = (UINT16)
336         ACPI_PTR_DIFF (ResourceSource, Descriptor);

338     DbgPrint (ASL_DEBUG_OUTPUT,
339         "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, IntLen: %
340         "GpioInt", Descriptor->Gpio.ResourceLength, (UINT16) sizeof (AML_RESOURC
341         ResSourceLength, VendorLength, InterruptLength);

343     /* Process all child initialization nodes */

345     for (i = 0; InitializerOp; i++)
346     {
347         switch (i)
348         {
349         case 0: /* Interrupt Mode - edge/level [Flag] (_MOD) */

351             RsSetFlagBits16 (&Descriptor->Gpio.IntFlags, InitializerOp, 0, 0);
352             RsCreateBitField (InitializerOp, ACPI_RESTAG_MODE,
353                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.IntFlags), 0);
354             break;

356         case 1: /* Interrupt Polarity - Active high/low [Flags] (_POL) */

358             RsSetFlagBits16 (&Descriptor->Gpio.IntFlags, InitializerOp, 1, 0);
359             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_POLARITY,
360                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.IntFlags), 1, 2);
361             break;

363         case 2: /* Share Type - Default: exclusive (0) [Flags] (_SHR) */

365             RsSetFlagBits16 (&Descriptor->Gpio.IntFlags, InitializerOp, 3, 0);
366             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_INTERRUPTSHARE,
367                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.IntFlags), 3, 2);
368             break;

370         case 3: /* Pin Config [BYTE] (_PPI) */

372             Descriptor->Gpio.PinConfig = (UINT8) InitializerOp->Asl.Value.Intege
373             RsCreateByteField (InitializerOp, ACPI_RESTAG_PINCONFIG,
374                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.PinConfig));
375             break;

377         case 4: /* Debounce Timeout [WORD] (_DBT) */

379             Descriptor->Gpio.DebounceTimeout = (UINT16) InitializerOp->Asl.Value
380             RsCreateWordField (InitializerOp, ACPI_RESTAG_DEBOUNCETIME,
381                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.DebounceTimeout));
382             break;

384         case 5: /* ResSource [Optional Field - STRING] */

386             if (ResSourceLength)
387             {
388                 /* Copy string to the descriptor */

390                 strcpy (ResourceSource,
391                     InitializerOp->Asl.Value.String);
```

```
392             }
393             break;

395         case 6: /* Resource Index */

397             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
398             {
399                 Descriptor->Gpio.ResSourceIndex = (UINT8) InitializerOp->Asl.Val
400             }
401             break;

403         case 7: /* Resource Usage (consumer/producer) */

405             RsSetFlagBits16 (&Descriptor->Gpio.Flags, InitializerOp, 0, 1);
406             break;

408         case 8: /* Resource Tag (Descriptor Name) */

410             UtAttachNamepathToOwner (Op, InitializerOp);
411             break;

413         case 9: /* Vendor Data (Optional - Buffer of BYTEs) (_VEN) */

415             /*
416              * Always set the VendorOffset even if there is no Vendor Data.
417              * This field is required in order to calculate the length
418              * of the ResourceSource at runtime.
419              */
420             Descriptor->Gpio.VendorOffset = (UINT16)
421                 ACPI_PTR_DIFF (VendorData, Descriptor);

423             if (RsGetVendorData (InitializerOp, VendorData,
424                     (CurrentByteOffset +  Descriptor->Gpio.VendorOffset)))
425             {
426                 Descriptor->Gpio.VendorLength = VendorLength;
427             }
428             break;

430         default:
431             /*
432              * PINs come through here, repeatedly. Each PIN must be a DWORD.
433              * NOTE: there is no "length" field for this, so from ACPI spec:
434              *  The number of pins in the table can be calculated from:
435              *  PinCount = (Resource Source Name Offset - Pin Table Offset) / 2
436              *  (implies resource source must immediately follow the pin list.)
437              *  Name: _PIN
438              */
439             *InterruptList = (UINT16) InitializerOp->Asl.Value.Integer;
440             InterruptList++;

442             /* Case 10: First interrupt number in list */

444             if (i == 10)
445             {
446                 if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
447                 {
448                     /* Must be at least one interrupt */

450                     AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_LIST_MIN,
451                         InitializerOp, NULL);
452                 }

454                 /* Check now for duplicates in list */

456                 RsCheckListForDuplicates (InitializerOp);
```



```
458                 /* Create a named field at the start of the list */

460                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_PIN,
461                     CurrentByteOffset + Descriptor->Gpio.PinTableOffset);
462             }
463             break;
464         }

466         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
467     }

469     return (Rnode);
470 }


473 /*******************************************************************************
474  *
475  * FUNCTION:    RsDoGpioIoDescriptor
476  *
477  * PARAMETERS:  Op                  - Parent resource descriptor parse node
478  *              CurrentByteOffset   - Offset into the resource template AML
479  *                                    buffer (to track references to the desc)
480  *
481  * RETURN:      Completed resource node
482  *
483  * DESCRIPTION: Construct a long "GpioIo" descriptor
484  *
485  ******************************************************************************/

487 ASL_RESOURCE_NODE *
488 RsDoGpioIoDescriptor (
489     ACPI_PARSE_OBJECT       *Op,
490     UINT32                  CurrentByteOffset)
491 {
492     AML_RESOURCE            *Descriptor;
493     ACPI_PARSE_OBJECT       *InitializerOp;
494     ASL_RESOURCE_NODE       *Rnode;
495     char                    *ResourceSource = NULL;
496     UINT8                   *VendorData = NULL;
497     UINT16                  *InterruptList = NULL;
498     UINT16                  ResSourceLength;
499     UINT16                  VendorLength;
500     UINT16                  InterruptLength;
501     UINT16                  DescriptorSize;
502     UINT32                  i;


505     InitializerOp = Op->Asl.Child;

507     /*
508      * Calculate lengths for fields that have variable length:
509      * 1) Resource Source string
510      * 2) Vendor Data buffer
511      * 3) PIN (interrupt) list
512      */
513     ResSourceLength = RsGetStringDataLength (InitializerOp);
514     VendorLength = RsGetBufferDataLength (InitializerOp);
515     InterruptLength = RsGetInterruptDataLength (InitializerOp);

517     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_GPIO) +
518         ResSourceLength + VendorLength + InterruptLength;

520     /* Allocate the local resource node and initialize */

522     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_
```

```
524     Descriptor = Rnode->Buffer;
525     Descriptor->Gpio.ResourceLength  = DescriptorSize;
526     Descriptor->Gpio.DescriptorType  = ACPI_RESOURCE_NAME_GPIO;
527     Descriptor->Gpio.RevisionId      = AML_RESOURCE_GPIO_REVISION;
528     Descriptor->Gpio.ConnectionType  = AML_RESOURCE_GPIO_TYPE_IO;

530     /* Build pointers to optional areas */

532     InterruptList = ACPI_ADD_PTR (UINT16, Descriptor, sizeof (AML_RESOURCE_GPIO)
533     ResourceSource = ACPI_ADD_PTR (char, InterruptList, InterruptLength);
534     VendorData = ACPI_ADD_PTR (UINT8, ResourceSource, ResSourceLength);

536     /* Setup offsets within the descriptor */

538     Descriptor->Gpio.PinTableOffset = (UINT16)
539         ACPI_PTR_DIFF (InterruptList, Descriptor);

541     Descriptor->Gpio.ResSourceOffset = (UINT16)
542         ACPI_PTR_DIFF (ResourceSource, Descriptor);

544     DbgPrint (ASL_DEBUG_OUTPUT,
545         "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, IntLen: %
546         "GpioIo", Descriptor->Gpio.ResourceLength, (UINT16) sizeof (AML_RESOURCE
547         ResSourceLength, VendorLength, InterruptLength);

549     /* Process all child initialization nodes */

551     for (i = 0; InitializerOp; i++)
552     {
553         switch (i)
554         {
555         case 0: /* Share Type [Flags] (_SHR) */

557             RsSetFlagBits16 (&Descriptor->Gpio.IntFlags, InitializerOp, 3, 0);
558             RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTSHARE,
559                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.IntFlags), 3);
560             break;

562         case 1: /* Pin Config [BYTE] (_PPI) */

564             Descriptor->Gpio.PinConfig = (UINT8) InitializerOp->Asl.Value.Intege
565             RsCreateByteField (InitializerOp, ACPI_RESTAG_PINCONFIG,
566                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.PinConfig));
567             break;

569         case 2: /* Debounce Timeout [WORD] (_DBT) */

571             Descriptor->Gpio.DebounceTimeout = (UINT16) InitializerOp->Asl.Value
572             RsCreateWordField (InitializerOp, ACPI_RESTAG_DEBOUNCETIME,
573                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.DebounceTimeout));
574             break;

576         case 3: /* Drive Strength [WORD] (_DRS) */

578             Descriptor->Gpio.DriveStrength = (UINT16) InitializerOp->Asl.Value.I
579             RsCreateWordField (InitializerOp, ACPI_RESTAG_DRIVESTRENGTH,
580                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.DriveStrength));
581             break;

583         case 4: /* I/O Restriction [Flag] (_IOR) */

585             RsSetFlagBits16 (&Descriptor->Gpio.IntFlags, InitializerOp, 0, 0);
586             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_IORESTRICTION,
587                 CurrentByteOffset + ASL_RESDESC_OFFSET (Gpio.IntFlags), 0, 2);
588             break;
```



```
590         case 5: /* ResSource [Optional Field - STRING] */

592             if (ResSourceLength)
593             {
594                 /* Copy string to the descriptor */

596                 strcpy (ResourceSource,
597                     InitializerOp->Asl.Value.String);
598             }
599             break;

601         case 6: /* Resource Index */

603             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
604             {
605                 Descriptor->Gpio.ResSourceIndex = (UINT8) InitializerOp->Asl.Val
606             }
607             break;

609         case 7: /* Resource Usage (consumer/producer) */

611             RsSetFlagBits16 (&Descriptor->Gpio.Flags, InitializerOp, 0, 1);
612             break;

614         case 8: /* Resource Tag (Descriptor Name) */

616             UtAttachNamepathToOwner (Op, InitializerOp);
617             break;

619         case 9: /* Vendor Data (Optional - Buffer of BYTEs) (_VEN) */
620             /*
621              * Always set the VendorOffset even if there is no Vendor Data.
622              * This field is required in order to calculate the length
623              * of the ResourceSource at runtime.
624              */
625             Descriptor->Gpio.VendorOffset = (UINT16)
626                 ACPI_PTR_DIFF (VendorData, Descriptor);

628             if (RsGetVendorData (InitializerOp, VendorData,
629                     (CurrentByteOffset + Descriptor->Gpio.VendorOffset)))
630             {
631                 Descriptor->Gpio.VendorLength = VendorLength;
632             }
633             break;

635         default:
636             /*
637              * PINs come through here, repeatedly. Each PIN must be a DWORD.
638              * NOTE: there is no "length" field for this, so from ACPI spec:
639              *  The number of pins in the table can be calculated from:
640              *  PinCount = (Resource Source Name Offset - Pin Table Offset) / 2
641              *  (implies resource source must immediately follow the pin list.)
642              *  Name: _PIN
643              */
644             *InterruptList = (UINT16) InitializerOp->Asl.Value.Integer;
645             InterruptList++;

647             /* Case 10: First interrupt number in list */

649             if (i == 10)
650             {
651                 if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
652                 {
653                     /* Must be at least one interrupt */

655                     AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_LIST_MIN,
```

```
656                         InitializerOp, NULL);
657                 }
659                 /* Check now for duplicates in list */
661                 RsCheckListForDuplicates (InitializerOp);
663                 /* Create a named field at the start of the list */
665                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_PIN,
666                     CurrentByteOffset + Descriptor->Gpio.PinTableOffset);
667             }
668             break;
669         }
671         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
672     }
674     return (Rnode);
675 }
678 /*******************************************************************************
679  *
680  * FUNCTION:    RsDoI2cSerialBusDescriptor
681  *
682  * PARAMETERS:  Op                  - Parent resource descriptor parse node
683  *              CurrentByteOffset   - Offset into the resource template AML
684  *                                    buffer (to track references to the desc)
685  *
686  * RETURN:      Completed resource node
687  *
688  * DESCRIPTION: Construct a long "I2cSerialBus" descriptor
689  *
690  ******************************************************************************/
692 ASL_RESOURCE_NODE *
693 RsDoI2cSerialBusDescriptor (
694     ACPI_PARSE_OBJECT       *Op,
695     UINT32                  CurrentByteOffset)
696 {
697     AML_RESOURCE            *Descriptor;
698     ACPI_PARSE_OBJECT       *InitializerOp;
699     ASL_RESOURCE_NODE       *Rnode;
700     char                    *ResourceSource = NULL;
701     UINT8                   *VendorData = NULL;
702     UINT16                  ResSourceLength;
703     UINT16                  VendorLength;
704     UINT16                  DescriptorSize;
705     UINT32                  i;
708     InitializerOp = Op->Asl.Child;
710     /*
711      * Calculate lengths for fields that have variable length:
712      * 1) Resource Source string
713      * 2) Vendor Data buffer
714      */
715     ResSourceLength = RsGetStringDataLength (InitializerOp);
716     VendorLength = RsGetBufferDataLength (InitializerOp);
718     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_I2C_SERIALBUS) +
719         ResSourceLength + VendorLength;
721     /* Allocate the local resource node and initialize */
```



```
723     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_
725     Descriptor = Rnode->Buffer;
726     Descriptor->I2cSerialBus.ResourceLength = DescriptorSize;
727     Descriptor->I2cSerialBus.DescriptorType = ACPI_RESOURCE_NAME_SERIAL_BUS;
728     Descriptor->I2cSerialBus.RevisionId     = AML_RESOURCE_I2C_REVISION;
729     Descriptor->I2cSerialBus.TypeRevisionId = AML_RESOURCE_I2C_TYPE_REVISION;
730     Descriptor->I2cSerialBus.Type           = AML_RESOURCE_I2C_SERIALBUSTYPE;
731     Descriptor->I2cSerialBus.TypeDataLength = AML_RESOURCE_I2C_MIN_DATA_LEN + Ve
733     /* Build pointers to optional areas */
735     VendorData = ACPI_ADD_PTR (UINT8, Descriptor, sizeof (AML_RESOURCE_I2C_SERIA
736     ResourceSource = ACPI_ADD_PTR (char, VendorData, VendorLength);
738     DbgPrint (ASL_DEBUG_OUTPUT,
739         "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, TypLen: %
740         "I2cSerialBus", Descriptor->I2cSerialBus.ResourceLength,
741         (UINT16) sizeof (AML_RESOURCE_I2C_SERIALBUS), ResSourceLength,
742         VendorLength, Descriptor->I2cSerialBus.TypeDataLength);
744     /* Process all child initialization nodes */
746     for (i = 0; InitializerOp; i++)
747     {
748         switch (i)
749         {
750         case 0: /* Slave Address [WORD] (_ADR) */
752             Descriptor->I2cSerialBus.SlaveAddress = (UINT16) InitializerOp->Asl.
753             RsCreateWordField (InitializerOp, ACPI_RESTAG_ADDRESS,
754                 CurrentByteOffset + ASL_RESDESC_OFFSET (I2cSerialBus.SlaveAddres
755             break;
757         case 1: /* Slave Mode [Flag] (_SLV) */
759             RsSetFlagBits (&Descriptor->I2cSerialBus.Flags, InitializerOp, 0, 0)
760             RsCreateBitField (InitializerOp, ACPI_RESTAG_SLAVEMODE,
761                 CurrentByteOffset + ASL_RESDESC_OFFSET (I2cSerialBus.Flags), 0);
762             break;
764         case 2: /* Connection Speed [DWORD] (_SPE) */
766             Descriptor->I2cSerialBus.ConnectionSpeed = (UINT32) InitializerOp->A
767             RsCreateDwordField (InitializerOp, ACPI_RESTAG_SPEED,
768                 CurrentByteOffset + ASL_RESDESC_OFFSET (I2cSerialBus.ConnectionS
769             break;
771         case 3: /* Addressing Mode [Flag] (_MOD) */
773             RsSetFlagBits16 (&Descriptor->I2cSerialBus.TypeSpecificFlags, Initia
774             RsCreateBitField (InitializerOp, ACPI_RESTAG_MODE,
775                 CurrentByteOffset + ASL_RESDESC_OFFSET (I2cSerialBus.TypeSpecifi
776             break;
778         case 4: /* ResSource [Optional Field - STRING] */
780             if (ResSourceLength)
781             {
782                 /* Copy string to the descriptor */
784                 strcpy (ResourceSource,
785                     InitializerOp->Asl.Value.String);
786             }
787             break;
```

```
789         case 5: /* Resource Index */

791             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
792             {
793                 Descriptor->I2cSerialBus.ResSourceIndex = (UINT8) InitializerOp-
794             }
795             break;

797         case 6: /* Resource Usage (consumer/producer) */

799             RsSetFlagBits (&Descriptor->I2cSerialBus.Flags, InitializerOp, 1, 1)
800             break;

802         case 7: /* Resource Tag (Descriptor Name) */

804             UtAttachNamepathToOwner (Op, InitializerOp);
805             break;

807         case 8: /* Vendor Data (Optional - Buffer of BYTEs) (_VEN) */

809             RsGetVendorData (InitializerOp, VendorData,
810                 CurrentByteOffset + sizeof (AML_RESOURCE_I2C_SERIALBUS));
811             break;

813         default:    /* Ignore any extra nodes */

815             break;
816         }

818         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
819     }

821     return (Rnode);
822 }


825 /*****************************************************************************
826  *
827  * FUNCTION:    RsDoSpiSerialBusDescriptor
828  *
829  * PARAMETERS:  Op                  - Parent resource descriptor parse node
830  *              CurrentByteOffset   - Offset into the resource template AML
831  *                                    buffer (to track references to the desc)
832  *
833  * RETURN:      Completed resource node
834  *
835  * DESCRIPTION: Construct a long "SPI Serial Bus" descriptor
836  *
837  *****************************************************************************/

839 ASL_RESOURCE_NODE *
840 RsDoSpiSerialBusDescriptor (
841     ACPI_PARSE_OBJECT       *Op,
842     UINT32                  CurrentByteOffset)
843 {
844     AML_RESOURCE            *Descriptor;
845     ACPI_PARSE_OBJECT       *InitializerOp;
846     ASL_RESOURCE_NODE       *Rnode;
847     char                    *ResourceSource = NULL;
848     UINT8                   *VendorData = NULL;
849     UINT16                  ResSourceLength;
850     UINT16                  VendorLength;
851     UINT16                  DescriptorSize;
852     UINT32                  i;
```



```
855     InitializerOp = Op->Asl.Child;

857     /*
858      * Calculate lengths for fields that have variable length:
859      * 1) Resource Source string
860      * 2) Vendor Data buffer
861      */
862     ResSourceLength = RsGetStringDataLength (InitializerOp);
863     VendorLength = RsGetBufferDataLength (InitializerOp);

865     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_SPI_SERIALBUS) +
866         ResSourceLength + VendorLength;

868     /* Allocate the local resource node and initialize */

870     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_

872     Descriptor = Rnode->Buffer;
873     Descriptor->SpiSerialBus.ResourceLength = DescriptorSize;
874     Descriptor->SpiSerialBus.DescriptorType = ACPI_RESOURCE_NAME_SERIAL_BUS;
875     Descriptor->SpiSerialBus.RevisionId     = AML_RESOURCE_SPI_REVISION;
876     Descriptor->SpiSerialBus.TypeRevisionId = AML_RESOURCE_SPI_TYPE_REVISION;
877     Descriptor->SpiSerialBus.Type           = AML_RESOURCE_SPI_SERIALBUSTYPE;
878     Descriptor->SpiSerialBus.TypeDataLength = AML_RESOURCE_SPI_MIN_DATA_LEN + Ve

880     /* Build pointers to optional areas */

882     VendorData = ACPI_ADD_PTR (UINT8, Descriptor, sizeof (AML_RESOURCE_SPI_SERIA
883     ResourceSource = ACPI_ADD_PTR (char, VendorData, VendorLength);

885     DbgPrint (ASL_DEBUG_OUTPUT,
886         "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, TypLen: %
887         "SpiSerialBus", Descriptor->SpiSerialBus.ResourceLength,
888         (UINT16) sizeof (AML_RESOURCE_SPI_SERIALBUS), ResSourceLength,
889         VendorLength, Descriptor->SpiSerialBus.TypeDataLength);

891     /* Process all child initialization nodes */

893     for (i = 0; InitializerOp; i++)
894     {
895         switch (i)
896         {
897         case 0: /* Device Selection [WORD] (_ADR) */

899             Descriptor->SpiSerialBus.DeviceSelection = (UINT16) InitializerOp->A
900             RsCreateWordField (InitializerOp, ACPI_RESTAG_ADDRESS,
901                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.DeviceSelec
902             break;

904         case 1: /* Device Polarity [Flag] (_DPL) */

906             RsSetFlagBits16 (&Descriptor->SpiSerialBus.TypeSpecificFlags, Initia
907             RsCreateBitField (InitializerOp, ACPI_RESTAG_DEVICEPOLARITY,
908                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.TypeSpecifi
909             break;

911         case 2: /* Wire Mode [Flag] (_MOD) */

913             RsSetFlagBits16 (&Descriptor->SpiSerialBus.TypeSpecificFlags, Initia
914             RsCreateBitField (InitializerOp, ACPI_RESTAG_MODE,
915                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.TypeSpecifi
916             break;

918         case 3: /* Device Bit Length [BYTE] (_LEN) */
```

```
920             Descriptor->SpiSerialBus.DataBitLength = (UINT8) InitializerOp->Asl.
921             RsCreateByteField (InitializerOp, ACPI_RESTAG_LENGTH,
922                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.DataBitLeng
923             break;
924
925         case 4: /* Slave Mode [Flag] (_SLV) */
926
927             RsSetFlagBits (&Descriptor->SpiSerialBus.Flags, InitializerOp, 0, 0)
928             RsCreateBitField (InitializerOp, ACPI_RESTAG_SLAVEMODE,
929                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.Flags), 0);
930             break;
931
932         case 5: /* Connection Speed [DWORD] (_SPE) */
933
934             Descriptor->SpiSerialBus.ConnectionSpeed = (UINT32) InitializerOp->A
935             RsCreateDwordField (InitializerOp, ACPI_RESTAG_SPEED,
936                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.ConnectionS
937             break;
938
939         case 6: /* Clock Polarity [BYTE] (_POL) */
940
941             Descriptor->SpiSerialBus.ClockPolarity = (UINT8) InitializerOp->Asl.
942             RsCreateByteField (InitializerOp, ACPI_RESTAG_POLARITY,
943                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.ClockPolari
944             break;
945
946         case 7: /* Clock Phase [BYTE] (_PHA) */
947
948             Descriptor->SpiSerialBus.ClockPhase = (UINT8) InitializerOp->Asl.Val
949             RsCreateByteField (InitializerOp, ACPI_RESTAG_PHASE,
950                 CurrentByteOffset + ASL_RESDESC_OFFSET (SpiSerialBus.ClockPhase)
951             break;
952
953         case 8: /* ResSource [Optional Field - STRING] */
954
955             if (ResSourceLength)
956             {
957                 /* Copy string to the descriptor */
958
959                 strcpy (ResourceSource,
960                     InitializerOp->Asl.Value.String);
961             }
962             break;
963
964         case 9: /* Resource Index */
965
966             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
967             {
968                 Descriptor->SpiSerialBus.ResSourceIndex = (UINT8) InitializerOp-
969             }
970             break;
971
972         case 10: /* Resource Usage (consumer/producer) */
973
974             RsSetFlagBits (&Descriptor->SpiSerialBus.Flags, InitializerOp, 1, 1)
975             break;
976
977         case 11: /* Resource Tag (Descriptor Name) */
978
979             UtAttachNamepathToOwner (Op, InitializerOp);
980             break;
981
982         case 12: /* Vendor Data (Optional - Buffer of BYTEs) (_VEN) */
983
984             RsGetVendorData (InitializerOp, VendorData,
985                 CurrentByteOffset + sizeof (AML_RESOURCE_SPI_SERIALBUS));
```



```
986             break;
987
988         default:    /* Ignore any extra nodes */
989
990             break;
991         }
992
993         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
994     }
995
996     return (Rnode);
997 }
998
999
1000 /*******************************************************************************
1001  *
1002  * FUNCTION:    RsDoUartSerialBusDescriptor
1003  *
1004  * PARAMETERS:  Op                  - Parent resource descriptor parse node
1005  *              CurrentByteOffset   - Offset into the resource template AML
1006  *                                    buffer (to track references to the desc)
1007  *
1008  * RETURN:      Completed resource node
1009  *
1010  * DESCRIPTION: Construct a long "UART Serial Bus" descriptor
1011  *
1012  ******************************************************************************/
1013
1014 ASL_RESOURCE_NODE *
1015 RsDoUartSerialBusDescriptor (
1016     ACPI_PARSE_OBJECT       *Op,
1017     UINT32                  CurrentByteOffset)
1018 {
1019     AML_RESOURCE            *Descriptor;
1020     ACPI_PARSE_OBJECT       *InitializerOp;
1021     ASL_RESOURCE_NODE       *Rnode;
1022     char                    *ResourceSource = NULL;
1023     UINT8                   *VendorData = NULL;
1024     UINT16                  ResSourceLength;
1025     UINT16                  VendorLength;
1026     UINT16                  DescriptorSize;
1027     UINT32                  i;
1028
1029
1030     InitializerOp = Op->Asl.Child;
1031
1032     /*
1033      * Calculate lengths for fields that have variable length:
1034      * 1) Resource Source string
1035      * 2) Vendor Data buffer
1036      */
1037     ResSourceLength = RsGetStringDataLength (InitializerOp);
1038     VendorLength = RsGetBufferDataLength (InitializerOp);
1039
1040     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_UART_SERIALBUS) +
1041         ResSourceLength + VendorLength;
1042
1043     /* Allocate the local resource node and initialize */
1044
1045     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_
1046
1047     Descriptor = Rnode->Buffer;
1048     Descriptor->UartSerialBus.ResourceLength = DescriptorSize;
1049     Descriptor->UartSerialBus.DescriptorType = ACPI_RESOURCE_NAME_SERIAL_BUS;
1050     Descriptor->UartSerialBus.RevisionId     = AML_RESOURCE_UART_REVISION;
1051     Descriptor->UartSerialBus.TypeRevisionId = AML_RESOURCE_UART_TYPE_REVISION;
```

```
    Descriptor->UartSerialBus.Type           = AML_RESOURCE_UART_SERIALBUSTYPE;
    Descriptor->UartSerialBus.TypeDataLength = AML_RESOURCE_UART_MIN_DATA_LEN +

    /* Build pointers to optional areas */

    VendorData = ACPI_ADD_PTR (UINT8, Descriptor, sizeof (AML_RESOURCE_UART_SERI
    ResourceSource = ACPI_ADD_PTR (char, VendorData, VendorLength);

    DbgPrint (ASL_DEBUG_OUTPUT,
        "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, TypLen: %
        "UartSerialBus", Descriptor->UartSerialBus.ResourceLength,
        (UINT16) sizeof (AML_RESOURCE_UART_SERIALBUS), ResSourceLength,
        VendorLength, Descriptor->UartSerialBus.TypeDataLength);

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Connection Speed (Baud Rate) [DWORD] (_SPE) */

            Descriptor->UartSerialBus.DefaultBaudRate = (UINT32) InitializerOp->
            RsCreateDwordField (InitializerOp, ACPI_RESTAG_SPEED,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.DefaultBau
            break;

        case 1: /* Bits Per Byte [Flags] (_LEN) */

            RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.TypeSpecif
            break;

        case 2: /* Stop Bits [Flags] (_STB) */

            RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_STOPBITS,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.TypeSpecif
            break;

        case 3: /* Lines In Use [BYTE] (_LIN) */

            Descriptor->UartSerialBus.LinesEnabled = (UINT8) InitializerOp->Asl.
            RsCreateByteField (InitializerOp, ACPI_RESTAG_LINE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.LinesEnabl
            break;

        case 4: /* Endianness [Flag] (_END) */

            RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
            RsCreateBitField (InitializerOp, ACPI_RESTAG_ENDIANNESS,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.TypeSpecif
            break;

        case 5: /* Parity [BYTE] (_PAR) */

            Descriptor->UartSerialBus.Parity = (UINT8) InitializerOp->Asl.Value.
            RsCreateByteField (InitializerOp, ACPI_RESTAG_PARITY,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.Parity));
            break;

        case 6: /* Flow Control [Flags] (_FLC) */

            RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
            RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_FLOWCONTROL,
```



```
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.TypeSpecif
            break;

        case 7: /* Rx Buffer Size [WORD] (_RXL) */

            Descriptor->UartSerialBus.RxFifoSize = (UINT16) InitializerOp->Asl.V
            RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH_RX,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.RxFifoSize
            break;

        case 8: /* Tx Buffer Size [WORD] (_TXL) */

            Descriptor->UartSerialBus.TxFifoSize = (UINT16) InitializerOp->Asl.V
            RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH_TX,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.TxFifoSize
            break;

        case 9: /* ResSource [Optional Field - STRING] */

            if (ResSourceLength)
            {
                /* Copy string to the descriptor */

                strcpy (ResourceSource,
                    InitializerOp->Asl.Value.String);
            }
            break;

        case 10: /* Resource Index */

            if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
            {
                Descriptor->UartSerialBus.ResSourceIndex = (UINT8) InitializerOp
            }
            break;

        case 11: /* Resource Usage (consumer/producer) */

            RsSetFlagBits (&Descriptor->UartSerialBus.Flags, InitializerOp, 1, 1

            /*
             * Slave Mode [Flag] (_SLV)
             *
             * Note: There is no SlaveMode argument to the UartSerialBus macro,
             * we add this name anyway to allow the flag to be set by ASL in the
             * rare case where there is a slave mode associated with the UART.
             */
            RsCreateBitField (InitializerOp, ACPI_RESTAG_SLAVEMODE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (UartSerialBus.Flags), 0)
            break;

        case 12: /* Resource Tag (Descriptor Name) */

            UtAttachNamepathToOwner (Op, InitializerOp);
            break;

        case 13: /* Vendor Data (Optional - Buffer of BYTEs) (_VEN) */

            RsGetVendorData (InitializerOp, VendorData,
                CurrentByteOffset + sizeof (AML_RESOURCE_UART_SERIALBUS));
            break;

        default:    /* Ignore any extra nodes */

            break;
        }
```

```
1185         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
1186     }

1188     return (Rnode);
1189 }
```

```
**********************************************************
   24719 Thu Dec 26 13:48:34 2013
new/usr/src/common/acpica/compiler/aslrestype2w.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslrestype2w - Large Word address resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslrestype2w")

/*
 * This module contains the Word (16-bit) address space descriptors:
 *
 * WordIO
 * WordMemory
 * WordSpace
 */

/******************************************************************************
 *
```



```
 * FUNCTION:    RsDoWordIoDescriptor
 *
 * PARAMETERS:  Op                  - Parent resource descriptor parse node
 *              CurrentByteOffset   - Offset into the resource template AML
 *                                    buffer (to track references to the desc)
 *
 * RETURN:      Completed resource node
 *
 * DESCRIPTION: Construct a long "WordIO" descriptor
 *
 *****************************************************************************/

ASL_RESOURCE_NODE *
RsDoWordIoDescriptor (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  CurrentByteOffset)
{
    AML_RESOURCE            *Descriptor;
    ACPI_PARSE_OBJECT       *InitializerOp;
    ACPI_PARSE_OBJECT       *MinOp = NULL;
    ACPI_PARSE_OBJECT       *MaxOp = NULL;
    ACPI_PARSE_OBJECT       *LengthOp = NULL;
    ACPI_PARSE_OBJECT       *GranOp = NULL;
    ASL_RESOURCE_NODE       *Rnode;
    UINT8                   *OptionalFields;
    UINT16                  StringLength = 0;
    UINT32                  OptionIndex = 0;
    UINT32                  i;
    BOOLEAN                 ResSourceIndex = FALSE;


    InitializerOp = Op->Asl.Child;
    StringLength = RsGetStringDataLength (InitializerOp);

    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->Address16.DescriptorType  = ACPI_RESOURCE_NAME_ADDRESS16;
    Descriptor->Address16.ResourceType    = ACPI_ADDRESS_TYPE_IO_RANGE;

    /*
     * Initial descriptor length -- may be enlarged if there are
     * optional fields present
     */
    OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS16);
    Descriptor->Address16.ResourceLength = (UINT16)
        (sizeof (AML_RESOURCE_ADDRESS16) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Usage */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 0, 1);
            break;

        case 1: /* MinType */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 2);
```

```
127             break;
129         case 2: /* MaxType */
131             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 3, 0);
132             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
133                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 3);
134             break;
136         case 3: /* DecodeType */
138             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 1, 0);
139             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
140                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 1);
141             break;
143         case 4: /* Range Type */
145             RsSetFlagBits (&Descriptor->Address16.SpecificFlags, InitializerOp,
146             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
147                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.SpecificFlags)
148             break;
150         case 5: /* Address Granularity */
152             Descriptor->Address16.Granularity = (UINT16) InitializerOp->Asl.Valu
153             RsCreateWordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
154                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Granularity));
155             GranOp = InitializerOp;
156             break;
158         case 6: /* Address Min */
160             Descriptor->Address16.Minimum = (UINT16) InitializerOp->Asl.Value.In
161             RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
162                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Minimum));
163             MinOp = InitializerOp;
164             break;
166         case 7: /* Address Max */
168             Descriptor->Address16.Maximum = (UINT16) InitializerOp->Asl.Value.In
169             RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
170                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Maximum));
171             MaxOp = InitializerOp;
172             break;
174         case 8: /* Translation Offset */
176             Descriptor->Address16.TranslationOffset = (UINT16) InitializerOp->As
177             RsCreateWordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
178                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.TranslationOff
179             break;
181         case 9: /* Address Length */
183             Descriptor->Address16.AddressLength = (UINT16) InitializerOp->Asl.Va
184             RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
185                  CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.AddressLength
186             LengthOp = InitializerOp;
187             break;
189         case 10: /* ResSourceIndex [Optional Field - BYTE] */
191             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
192             {
```



```
193                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
194                 OptionIndex++;
195                 Descriptor->Address16.ResourceLength++;
196                 ResSourceIndex = TRUE;
197             }
198             break;
200         case 11: /* ResSource [Optional Field - STRING] */
202             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
203                 (InitializerOp->Asl.Value.String))
204             {
205                 if (StringLength)
206                 {
207                     Descriptor->Address16.ResourceLength = (UINT16)
208                         (Descriptor->Address16.ResourceLength + StringLength);
210                     strcpy ((char *)
211                         &OptionalFields[OptionIndex],
212                         InitializerOp->Asl.Value.String);
214                     /* ResourceSourceIndex must also be valid */
216                     if (!ResSourceIndex)
217                     {
218                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
219                             InitializerOp, NULL);
220                     }
221                 }
222             }
224 #if 0
225             /*
226              * Not a valid ResourceSource, ResourceSourceIndex must also
227              * be invalid
228              */
229             else if (ResSourceIndex)
230             {
231                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
232                     InitializerOp, NULL);
233             }
234 #endif
235             break;
237         case 12: /* ResourceTag */
239             UtAttachNamepathToOwner (Op, InitializerOp);
240             break;
242         case 13: /* Type */
244             RsSetFlagBits (&Descriptor->Address16.SpecificFlags, InitializerOp,
245             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
246                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.SpecificFlags)
247             break;
249         case 14: /* Translation Type */
251             RsSetFlagBits (&Descriptor->Address16.SpecificFlags, InitializerOp,
252             RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
253                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.SpecificFlags)
254             break;
256         default:
258             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
```

```
259             break;
260         }
261
262         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
263     }
264
265     /* Validate the Min/Max/Len/Gran values */
266
267     RsLargeAddressCheck (
268         (UINT64) Descriptor->Address16.Minimum,
269         (UINT64) Descriptor->Address16.Maximum,
270         (UINT64) Descriptor->Address16.AddressLength,
271         (UINT64) Descriptor->Address16.Granularity,
272         Descriptor->Address16.Flags,
273         MinOp, MaxOp, LengthOp, GranOp, Op);
274
275     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS16) +
276                           OptionIndex + StringLength;
277     return (Rnode);
278 }
279
280
281 /*******************************************************************************
282  *
283  * FUNCTION:    RsDoWordBusNumberDescriptor
284  *
285  * PARAMETERS:  Op                  - Parent resource descriptor parse node
286  *              CurrentByteOffset   - Offset into the resource template AML
287  *                                    buffer (to track references to the desc)
288  *
289  * RETURN:      Completed resource node
290  *
291  * DESCRIPTION: Construct a long "WordBusNumber" descriptor
292  *
293  ******************************************************************************/
294
295 ASL_RESOURCE_NODE *
296 RsDoWordBusNumberDescriptor (
297     ACPI_PARSE_OBJECT       *Op,
298     UINT32                  CurrentByteOffset)
299 {
300     AML_RESOURCE            *Descriptor;
301     ACPI_PARSE_OBJECT       *InitializerOp;
302     ACPI_PARSE_OBJECT       *MinOp = NULL;
303     ACPI_PARSE_OBJECT       *MaxOp = NULL;
304     ACPI_PARSE_OBJECT       *LengthOp = NULL;
305     ACPI_PARSE_OBJECT       *GranOp = NULL;
306     ASL_RESOURCE_NODE       *Rnode;
307     UINT8                   *OptionalFields;
308     UINT16                  StringLength = 0;
309     UINT32                  OptionIndex = 0;
310     UINT32                  i;
311     BOOLEAN                 ResSourceIndex = FALSE;
312
313
314     InitializerOp = Op->Asl.Child;
315     StringLength = RsGetStringDataLength (InitializerOp);
316
317     Rnode = RsAllocateResourceNode (
318                 sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);
319
320     Descriptor = Rnode->Buffer;
321     Descriptor->Address16.DescriptorType  = ACPI_RESOURCE_NAME_ADDRESS16;
322     Descriptor->Address16.ResourceType    = ACPI_ADDRESS_TYPE_BUS_NUMBER_RANGE;
323
324     /*
```



```
325      * Initial descriptor length -- may be enlarged if there are
326      * optional fields present
327      */
328     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS16);
329     Descriptor->Address16.ResourceLength = (UINT16)
330         (sizeof (AML_RESOURCE_ADDRESS16) -
331          sizeof (AML_RESOURCE_LARGE_HEADER));
332
333     /* Process all child initialization nodes */
334
335     for (i = 0; InitializerOp; i++)
336     {
337         switch (i)
338         {
339         case 0: /* Resource Usage */
340
341             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 0, 1);
342             break;
343
344         case 1: /* MinType */
345
346             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
347             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
348                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 2);
349             break;
350
351         case 2: /* MaxType */
352
353             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 3, 0);
354             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
355                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 3);
356             break;
357
358         case 3: /* DecodeType */
359
360             RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 1, 0);
361             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
362                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 1);
363             break;
364
365         case 4: /* Address Granularity */
366
367             Descriptor->Address16.Granularity =
368                 (UINT16) InitializerOp->Asl.Value.Integer;
369             RsCreateWordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
370                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Granularity));
371             GranOp = InitializerOp;
372             break;
373
374         case 5: /* Min Address */
375
376             Descriptor->Address16.Minimum =
377                 (UINT16) InitializerOp->Asl.Value.Integer;
378             RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
379                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Minimum));
380             MinOp = InitializerOp;
381             break;
382
383         case 6: /* Max Address */
384
385             Descriptor->Address16.Maximum =
386                 (UINT16) InitializerOp->Asl.Value.Integer;
387             RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
388                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Maximum));
389             MaxOp = InitializerOp;
390             break;
```

```
392         case 7: /* Translation Offset */

394             Descriptor->Address16.TranslationOffset =
395                 (UINT16) InitializerOp->Asl.Value.Integer;
396             RsCreateWordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
397                 CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.TranslationOff
398             break;

400         case 8: /* Address Length */

402             Descriptor->Address16.AddressLength =
403                 (UINT16) InitializerOp->Asl.Value.Integer;
404             RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
405                  CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.AddressLength
406             LengthOp = InitializerOp;
407             break;

409         case 9: /* ResSourceIndex [Optional Field - BYTE] */

411             if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
412             {
413                 OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
414                 OptionIndex++;
415                 Descriptor->Address16.ResourceLength++;
416                 ResSourceIndex = TRUE;
417             }
418             break;

420         case 10: /* ResSource [Optional Field - STRING] */

422             if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
423                 (InitializerOp->Asl.Value.String))
424             {
425                 if (StringLength)
426                 {
427                     Descriptor->Address16.ResourceLength = (UINT16)
428                         (Descriptor->Address16.ResourceLength + StringLength);

430                     strcpy ((char *)
431                         &OptionalFields[OptionIndex],
432                         InitializerOp->Asl.Value.String);

434                     /* ResourceSourceIndex must also be valid */

436                     if (!ResSourceIndex)
437                     {
438                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
439                             InitializerOp, NULL);
440                     }
441                 }
442             }

444 #if 0
445             /*
446              * Not a valid ResourceSource, ResourceSourceIndex must also
447              * be invalid
448              */
449             else if (ResSourceIndex)
450             {
451                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
452                     InitializerOp, NULL);
453             }
454 #endif
455             break;
```



```
457         case 11: /* ResourceTag */

459             UtAttachNamepathToOwner (Op, InitializerOp);
460             break;

462         default:

464             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
465             break;
466         }

468         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
469     }

471     /* Validate the Min/Max/Len/Gran values */

473     RsLargeAddressCheck (
474         (UINT64) Descriptor->Address16.Minimum,
475         (UINT64) Descriptor->Address16.Maximum,
476         (UINT64) Descriptor->Address16.AddressLength,
477         (UINT64) Descriptor->Address16.Granularity,
478         Descriptor->Address16.Flags,
479         MinOp, MaxOp, LengthOp, GranOp, Op);

481     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS16) +
482                             OptionIndex + StringLength;
483     return (Rnode);
484 }


487 /*******************************************************************************
488  *
489  * FUNCTION:    RsDoWordSpaceDescriptor
490  *
491  * PARAMETERS:  Op                  - Parent resource descriptor parse node
492  *              CurrentByteOffset   - Offset into the resource template AML
493  *                                    buffer (to track references to the desc)
494  *
495  * RETURN:      Completed resource node
496  *
497  * DESCRIPTION: Construct a long "WordSpace" descriptor
498  *
499  ******************************************************************************/

501 ASL_RESOURCE_NODE *
502 RsDoWordSpaceDescriptor (
503     ACPI_PARSE_OBJECT       *Op,
504     UINT32                  CurrentByteOffset)
505 {
506     AML_RESOURCE            *Descriptor;
507     ACPI_PARSE_OBJECT       *InitializerOp;
508     ACPI_PARSE_OBJECT       *MinOp = NULL;
509     ACPI_PARSE_OBJECT       *MaxOp = NULL;
510     ACPI_PARSE_OBJECT       *LengthOp = NULL;
511     ACPI_PARSE_OBJECT       *GranOp = NULL;
512     ASL_RESOURCE_NODE       *Rnode;
513     UINT8                   *OptionalFields;
514     UINT16                  StringLength = 0;
515     UINT32                  OptionIndex = 0;
516     UINT32                  i;
517     BOOLEAN                 ResSourceIndex = FALSE;


520     InitializerOp = Op->Asl.Child;
521     StringLength = RsGetStringDataLength (InitializerOp);
```

```
    Rnode = RsAllocateResourceNode (
                sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);

    Descriptor = Rnode->Buffer;
    Descriptor->Address16.DescriptorType  = ACPI_RESOURCE_NAME_ADDRESS16;

    /*
     * Initial descriptor length -- may be enlarged if there are
     * optional fields present
     */
    OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS16);
    Descriptor->Address16.ResourceLength = (UINT16)
        (sizeof (AML_RESOURCE_ADDRESS16) -
         sizeof (AML_RESOURCE_LARGE_HEADER));

    /* Process all child initialization nodes */

    for (i = 0; InitializerOp; i++)
    {
        switch (i)
        {
        case 0: /* Resource Type */

            Descriptor->Address16.ResourceType =
                (UINT8) InitializerOp->Asl.Value.Integer;
            break;

        case 1: /* Resource Usage */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 0, 1);
            break;

        case 2: /* DecodeType */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 1, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 1);
            break;

        case 3: /* MinType */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 2);
            break;

        case 4: /* MaxType */

            RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 3, 0);
            RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Flags), 3);
            break;

        case 5: /* Type-Specific flags */

            Descriptor->Address16.SpecificFlags =
                (UINT8) InitializerOp->Asl.Value.Integer;
            break;

        case 6: /* Address Granularity */

            Descriptor->Address16.Granularity =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Granularity));
            GranOp = InitializerOp;
```



```
            break;

        case 7: /* Min Address */

            Descriptor->Address16.Minimum =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Minimum));
            MinOp = InitializerOp;
            break;

        case 8: /* Max Address */

            Descriptor->Address16.Maximum =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.Maximum));
            MaxOp = InitializerOp;
            break;

        case 9: /* Translation Offset */

            Descriptor->Address16.TranslationOffset =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.TranslationOff
            break;

        case 10: /* Address Length */

            Descriptor->Address16.AddressLength =
                (UINT16) InitializerOp->Asl.Value.Integer;
            RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
                CurrentByteOffset + ASL_RESDESC_OFFSET (Address16.AddressLength)
            LengthOp = InitializerOp;
            break;

        case 11: /* ResSourceIndex [Optional Field - BYTE] */

            if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
            {
                OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
                OptionIndex++;
                Descriptor->Address16.ResourceLength++;
                ResSourceIndex = TRUE;
            }
            break;

        case 12: /* ResSource [Optional Field - STRING] */

            if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
                (InitializerOp->Asl.Value.String))
            {
                if (StringLength)
                {
                    Descriptor->Address16.ResourceLength = (UINT16)
                        (Descriptor->Address16.ResourceLength + StringLength);

                    strcpy ((char *)
                        &OptionalFields[OptionIndex],
                        InitializerOp->Asl.Value.String);

                    /* ResourceSourceIndex must also be valid */

                    if (!ResSourceIndex)
                    {
```

```
655                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
656                             InitializerOp, NULL);
657                     }
658                 }
659             }

661 #if 0
662             /*
663              * Not a valid ResourceSource, ResourceSourceIndex must also
664              * be invalid
665              */
666             else if (ResSourceIndex)
667             {
668                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
669                     InitializerOp, NULL);
670             }
671 #endif
672             break;

674         case 13: /* ResourceTag */

676             UtAttachNamepathToOwner (Op, InitializerOp);
677             break;

679         default:

681             AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
682             break;
683         }

685         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
686     }

688     /* Validate the Min/Max/Len/Gran values */

690     RsLargeAddressCheck (
691         (UINT64) Descriptor->Address16.Minimum,
692         (UINT64) Descriptor->Address16.Maximum,
693         (UINT64) Descriptor->Address16.AddressLength,
694         (UINT64) Descriptor->Address16.Granularity,
695         Descriptor->Address16.Flags,
696         MinOp, MaxOp, LengthOp, GranOp, Op);

698     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS16) +
699                           OptionIndex + StringLength;
700     return (Rnode);
701 }
```

```
**********************************************************
   13219 Thu Dec 26 13:48:34 2013
new/usr/src/common/acpica/compiler/aslstartup.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: aslstartup - Compiler startup routines, called from main
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #include "aslcompiler.h"
  46 #include "actables.h"
  47 #include "acdisasm.h"
  48 #include "acapps.h"

  50 #define _COMPONENT          ACPI_COMPILER
  51         ACPI_MODULE_NAME    ("aslstartup")


  54 /* Local prototypes */

  56 static UINT8
  57 AslDetectSourceFileType (
  58     ASL_FILE_INFO           *Info);
```



```
  60 static ACPI_STATUS
  61 AslDoDisassembly (
  62     void);


  65 /* Globals */

  67 static BOOLEAN          AslToFile = TRUE;


  70 /******************************************************************************
  71  *
  72  * FUNCTION:    AslInitializeGlobals
  73  *
  74  * PARAMETERS:  None
  75  *
  76  * RETURN:      None
  77  *
  78  * DESCRIPTION: Re-initialize globals needed to restart the compiler. This
  79  *              allows multiple files to be disassembled and/or compiled.
  80  *
  81  *****************************************************************************/

  83 void
  84 AslInitializeGlobals (
  85     void)
  86 {
  87     UINT32                  i;


  90     /* Init compiler globals */

  92     Gbl_CurrentColumn = 0;
  93     Gbl_CurrentLineNumber = 1;
  94     Gbl_LogicalLineNumber = 1;
  95     Gbl_CurrentLineOffset = 0;
  96     Gbl_InputFieldCount = 0;
  97     Gbl_InputByteCount = 0;
  98     Gbl_NsLookupCount = 0;
  99     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;

 101     Gbl_ErrorLog = NULL;
 102     Gbl_NextError = NULL;
 103     Gbl_Signature = NULL;
 104     Gbl_FileType = 0;

 106     TotalExecutableOpcodes = 0;
 107     TotalNamedObjects = 0;
 108     TotalKeywords = 0;
 109     TotalParseNodes = 0;
 110     TotalMethods = 0;
 111     TotalAllocations = 0;
 112     TotalAllocated = 0;
 113     TotalFolds = 0;

 115     AslGbl_NextEvent = 0;
 116     for (i = 0; i < ASL_NUM_REPORT_LEVELS; i++)
 117     {
 118         Gbl_ExceptionCount[i] = 0;
 119     }

 121     for (i = ASL_FILE_INPUT; i <= ASL_MAX_FILE_TYPE; i++)
 122     {
 123         Gbl_Files[i].Handle = NULL;
 124         Gbl_Files[i].Filename = NULL;
```

```
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AslDetectSourceFileType
 *
 * PARAMETERS:  Info            - Name/Handle for the file (must be open)
 *
 * RETURN:      File Type
 *
 * DESCRIPTION: Determine the type of the input file. Either binary (contains
 *              non-ASCII characters), ASL file, or an ACPI Data Table file.
 *
 ******************************************************************************/

static UINT8
AslDetectSourceFileType (
    ASL_FILE_INFO           *Info)
{
    char                    *FileChar;
    UINT8                   Type;
    ACPI_STATUS             Status;


    /* Check for a valid binary ACPI table */

    Status = FlCheckForAcpiTable (Info->Handle);
    if (ACPI_SUCCESS (Status))
    {
        Type = ASL_INPUT_TYPE_ACPI_TABLE;
        goto Cleanup;
    }

    /* Check for 100% ASCII source file (comments are ignored) */

    Status = FlCheckForAscii (Info->Handle, Info->Filename, TRUE);
    if (ACPI_FAILURE (Status))
    {
        printf ("Non-ascii input file - %s\n", Info->Filename);

        if (!Gbl_IgnoreErrors)
        {
            Type = ASL_INPUT_TYPE_BINARY;
            goto Cleanup;
        }
    }

    /*
     * File is ASCII. Determine if this is an ASL file or an ACPI data
     * table file.
     */
    while (fgets (Gbl_CurrentLineBuffer, Gbl_LineBufferSize, Info->Handle))
    {
        /* Uppercase the buffer for caseless compare */

        FileChar = Gbl_CurrentLineBuffer;
        while (*FileChar)
        {
            *FileChar = (char) toupper ((int) *FileChar);
            FileChar++;
        }

        /* Presence of "DefinitionBlock" indicates actual ASL code */
```



```
        if (strstr (Gbl_CurrentLineBuffer, "DEFINITIONBLOCK"))
        {
            /* Appears to be an ASL file */

            Type = ASL_INPUT_TYPE_ASCII_ASL;
            goto Cleanup;
        }
    }

    /* Not an ASL source file, default to a data table source file */

    Type = ASL_INPUT_TYPE_ASCII_DATA;

Cleanup:

    /* Must seek back to the start of the file */

    fseek (Info->Handle, 0, SEEK_SET);
    return (Type);
}


/*******************************************************************************
 *
 * FUNCTION:    AslDoDisassembly
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initiate AML file disassembly. Uses ACPICA subsystem to build
 *              namespace.
 *
 ******************************************************************************/

static ACPI_STATUS
AslDoDisassembly (
    void)
{
    ACPI_STATUS             Status;


    /* ACPICA subsystem initialization */

    Status = AdInitialize ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    Status = AcpiAllocateRootTable (4);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("Could not initialize ACPI Table Manager, %s\n",
            AcpiFormatException (Status));
        return (Status);
    }

    /* This is where the disassembly happens */

    AcpiGbl_DbOpt_disasm = TRUE;
    Status = AdAmlDisassemble (AslToFile,
        Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_OutputFilenamePrefix,
        &Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_GetAllTables);
    if (ACPI_FAILURE (Status))
    {
```

```
257         return (Status);
258     }

260     /* Check if any control methods were unresolved */

262     AcpiDmUnresolvedWarning (0);

264 #if 0
265     /* TBD: Handle additional output files for disassembler */

267     Status = FlOpenMiscOutputFiles (Gbl_OutputFilenamePrefix);
268     NsDisplayNamespace ();
269 #endif

271     /* Shutdown compiler and ACPICA subsystem */

273     AeClearErrorLog ();
274     (void) AcpiTerminate ();

276     /*
277      * Gbl_Files[ASL_FILE_INPUT].Filename was replaced with the
278      * .DSL disassembly file, which can now be compiled if requested
279      */
280     if (Gbl_DoCompile)
281     {
282         AcpiOsPrintf ("\nCompiling \"%s\"\n",
283             Gbl_Files[ASL_FILE_INPUT].Filename);
284         return (AE_CTRL_CONTINUE);
285     }

287     ACPI_FREE (Gbl_Files[ASL_FILE_INPUT].Filename);
288     Gbl_Files[ASL_FILE_INPUT].Filename = NULL;
289     return (AE_OK);
290 }


293 /*******************************************************************************
294  *
295  * FUNCTION:    AslDoOneFile
296  *
297  * PARAMETERS:  Filename        - Name of the file
298  *
299  * RETURN:      Status
300  *
301  * DESCRIPTION: Process a single file - either disassemble, compile, or both
302  *
303  ******************************************************************************/

305 ACPI_STATUS
306 AslDoOneFile (
307     char                    *Filename)
308 {
309     ACPI_STATUS             Status;


312     /* Re-initialize "some" compiler/preprocessor globals */

314     AslInitializeGlobals ();
315     PrInitializeGlobals ();

317     /*
318      * Extract the directory path. This path is used for possible include
319      * files and the optional AML filename embedded in the input file
320      * DefinitionBlock declaration.
321      */
322     Status = FlSplitInputPathname (Filename, &Gbl_DirectoryPath, NULL);
```



```
323     if (ACPI_FAILURE (Status))
324     {
325         return (Status);
326     }

328     Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
329     UtConvertBackslashes (Filename);

331     /*
332      * AML Disassembly (Optional)
333      */
334     if (Gbl_DisasmFlag || Gbl_GetAllTables)
335     {
336         Status = AslDoDisassembly ();
337         if (Status != AE_CTRL_CONTINUE)
338         {
339             return (Status);
340         }
341     }

343     /*
344      * Open the input file. Here, this should be an ASCII source file,
345      * either an ASL file or a Data Table file
346      */
347     Status = FlOpenInputFile (Gbl_Files[ASL_FILE_INPUT].Filename);
348     if (ACPI_FAILURE (Status))
349     {
350         AePrintErrorLog (ASL_FILE_STDERR);
351         return (AE_ERROR);
352     }

354     /* Determine input file type */

356     Gbl_FileType = AslDetectSourceFileType (&Gbl_Files[ASL_FILE_INPUT]);
357     if (Gbl_FileType == ASL_INPUT_TYPE_BINARY)
358     {
359         return (AE_ERROR);
360     }

362     /*
363      * If -p not specified, we will use the input filename as the
364      * output filename prefix
365      */
366     if (Gbl_UseDefaultAmlFilename)
367     {
368         Gbl_OutputFilenamePrefix = Gbl_Files[ASL_FILE_INPUT].Filename;
369     }

371     /* Open the optional output files (listings, etc.) */

373     Status = FlOpenMiscOutputFiles (Gbl_OutputFilenamePrefix);
374     if (ACPI_FAILURE (Status))
375     {
376         AePrintErrorLog (ASL_FILE_STDERR);
377         return (AE_ERROR);
378     }

380     /*
381      * Compilation of ASL source versus DataTable source uses different
382      * compiler subsystems
383      */
384     switch (Gbl_FileType)
385     {
386     /*
387      * Data Table Compilation
388      */
```

```
389     case ASL_INPUT_TYPE_ASCII_DATA:

391         Status = DtDoCompile ();
392         if (ACPI_FAILURE (Status))
393         {
394             return (Status);
395         }

397         if (Gbl_Signature)
398         {
399             ACPI_FREE (Gbl_Signature);
400             Gbl_Signature = NULL;
401         }

403         /* Check if any errors occurred during compile */

405         Status = AslCheckForErrorExit ();
406         if (ACPI_FAILURE (Status))
407         {
408             return (Status);
409         }

411         /* Cleanup (for next source file) and exit */

413         AeClearErrorLog ();
414         PrTerminatePreprocessor ();
415         return (Status);

417     /*
418      * ASL Compilation
419      */
420     case ASL_INPUT_TYPE_ASCII_ASL:

422         /* ACPICA subsystem initialization */

424         Status = AdInitialize ();
425         if (ACPI_FAILURE (Status))
426         {
427             return (Status);
428         }

430         (void) CmDoCompile ();
431         (void) AcpiTerminate ();

433         /* Check if any errors occurred during compile */

435         Status = AslCheckForErrorExit ();
436         if (ACPI_FAILURE (Status))
437         {
438             return (Status);
439         }

441         /* Cleanup (for next source file) and exit */

443         AeClearErrorLog ();
444         PrTerminatePreprocessor ();
445         return (AE_OK);

447     /*
448      * Binary ACPI table was auto-detected, disassemble it
449      */
450     case ASL_INPUT_TYPE_ACPI_TABLE:

452         /* We have what appears to be an ACPI table, disassemble it */

454         FlCloseFile (ASL_FILE_INPUT);
```



```
455         Gbl_DoCompile = FALSE;
456         Gbl_DisasmFlag = TRUE;
457         Status = AslDoDisassembly ();
458         return (Status);

460     /* Unknown binary table */

462     case ASL_INPUT_TYPE_BINARY:

464         AePrintErrorLog (ASL_FILE_STDERR);
465         return (AE_ERROR);

467     default:

469         printf ("Unknown file type %X\n", Gbl_FileType);
470         return (AE_ERROR);
471     }
472 }


475 /*******************************************************************************
476  *
477  * FUNCTION:    AslCheckForErrorExit
478  *
479  * PARAMETERS:  None. Examines global exception count array
480  *
481  * RETURN:      Status
482  *
483  * DESCRIPTION: Determine if compiler should abort with error status
484  *
485  ******************************************************************************/

487 ACPI_STATUS
488 AslCheckForErrorExit (
489     void)
490 {

492     /*
493      * Return non-zero exit code if there have been errors, unless the
494      * global ignore error flag has been set
495      */
496     if (!Gbl_IgnoreErrors)
497     {
498         if (Gbl_ExceptionCount[ASL_ERROR] > 0)
499         {
500             return (AE_ERROR);
501         }

503         /* Optionally treat warnings as errors */

505         if (Gbl_WarningsAsErrors)
506         {
507             if ((Gbl_ExceptionCount[ASL_WARNING] > 0)  ||
508                 (Gbl_ExceptionCount[ASL_WARNING2] > 0) ||
509                 (Gbl_ExceptionCount[ASL_WARNING3] > 0))
510             {
511                 return (AE_ERROR);
512             }
513         }
514     }

516     return (AE_OK);
517 }
```

```
**********************************************************
    5855 Thu Dec 26 13:48:34 2013
new/usr/src/common/acpica/compiler/aslstubs.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslstubs - Stubs used to link to Aml interpreter
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "acdispat.h"
#include "actables.h"
#include "acevents.h"
#include "acinterp.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslstubs")


/*
 * Stubs to simplify linkage to the ACPI CA core subsystem.
 * Things like Events, Global Lock, etc. are not used
 * by the compiler, so they are stubbed out here.
 */
```



```
ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void)
{
    return (0);
}

void
AcpiNsExecModuleCodeList (
    void)
{
}

ACPI_STATUS
AcpiHwReadPort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  *Value,
    UINT32                  Width)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiHwWritePort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  Value,
    UINT32                  Width)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsMethodError (
    ACPI_STATUS             Status,
    ACPI_WALK_STATE         *WalkState)
{
    return (Status);
}

ACPI_STATUS
AcpiDsMethodDataGetValue (
    UINT8                   Type,
    UINT32                  Index,
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     **DestDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsMethodDataGetNode (
    UINT8                   Type,
    UINT32                  Index,
    ACPI_WALK_STATE         *WalkState,
    ACPI_NAMESPACE_NODE     **Node)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsStoreObjectToLocal (
    UINT8                   Type,
    UINT32                  Index,
    ACPI_OPERAND_OBJECT     *SrcDesc,
    ACPI_WALK_STATE         *WalkState)
{
```

```
126     return (AE_OK);
127 }

129 ACPI_STATUS
130 AcpiEvQueueNotifyRequest (
131     ACPI_NAMESPACE_NODE     *Node,
132     UINT32                  NotifyValue)
133 {
134     return (AE_OK);
135 }

137 BOOLEAN
138 AcpiEvIsNotifyObject (
139     ACPI_NAMESPACE_NODE     *Node)
140 {
141     return (FALSE);
142 }

144 #if (!ACPI_REDUCED_HARDWARE)
145 ACPI_STATUS
146 AcpiEvDeleteGpeBlock (
147     ACPI_GPE_BLOCK_INFO     *GpeBlock)
148 {
149     return (AE_OK);
150 }

152 ACPI_STATUS
153 AcpiEvAcquireGlobalLock (
154     UINT16                  Timeout)
155 {
156     return (AE_OK);
157 }

159 ACPI_STATUS
160 AcpiEvReleaseGlobalLock (
161     void)
162 {
163     return (AE_OK);
164 }
165 #endif /* !ACPI_REDUCED_HARDWARE */

167 ACPI_STATUS
168 AcpiEvInitializeRegion (
169     ACPI_OPERAND_OBJECT     *RegionObj,
170     BOOLEAN                 AcpiNsLocked)
171 {
172     return (AE_OK);
173 }

175 void
176 AcpiExDoDebugObject (
177     ACPI_OPERAND_OBJECT     *SourceDesc,
178     UINT32                  Level,
179     UINT32                  Index)
180 {
181     return;
182 }

184 ACPI_STATUS
185 AcpiExReadDataFromField (
186     ACPI_WALK_STATE         *WalkState,
187     ACPI_OPERAND_OBJECT     *ObjDesc,
188     ACPI_OPERAND_OBJECT     **RetBufferDesc)
189 {
190     return (AE_SUPPORT);
191 }
```



```
193 ACPI_STATUS
194 AcpiExWriteDataToField (
195     ACPI_OPERAND_OBJECT     *SourceDesc,
196     ACPI_OPERAND_OBJECT     *ObjDesc,
197     ACPI_OPERAND_OBJECT     **ResultDesc)
198 {
199     return (AE_SUPPORT);
200 }

202 ACPI_STATUS
203 AcpiExLoadTableOp (
204     ACPI_WALK_STATE         *WalkState,
205     ACPI_OPERAND_OBJECT     **ReturnDesc)
206 {
207     return (AE_SUPPORT);
208 }

210 ACPI_STATUS
211 AcpiExUnloadTable (
212     ACPI_OPERAND_OBJECT     *DdbHandle)
213 {
214     return (AE_SUPPORT);
215 }

217 ACPI_STATUS
218 AcpiExLoadOp (
219     ACPI_OPERAND_OBJECT     *ObjDesc,
220     ACPI_OPERAND_OBJECT     *Target,
221     ACPI_WALK_STATE         *WalkState)
222 {
223     return (AE_SUPPORT);
224 }

226 ACPI_STATUS
227 AcpiTbFindTable (
228     char                    *Signature,
229     char                    *OemId,
230     char                    *OemTableId,
231     UINT32                  *TableIndex)
232 {
233     return (AE_SUPPORT);
234 }

236 /* OSL interfaces */

238 ACPI_THREAD_ID
239 AcpiOsGetThreadId (
240     void)
241 {
242     return (1);
243 }

245 ACPI_STATUS
246 AcpiOsExecute (
247     ACPI_EXECUTE_TYPE       Type,
248     ACPI_OSD_EXEC_CALLBACK  Function,
249     void                    *Context)
250 {
251     return (AE_SUPPORT);
252 }
```

```
**********************************************************
   21957 Thu Dec 26 13:48:35 2013
new/usr/src/common/acpica/compiler/aslsupport.l
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslsupport.l - Flex/lex scanner C support routines.
 *              NOTE: Included into aslcompile.l, not compiled by itself.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


/* Configuration */

#define ASL_SPACES_PER_TAB      4

#define ASL_NORMAL_CHAR         0
#define ASL_ESCAPE_SEQUENCE     1
#define ASL_OCTAL_CONSTANT      2
#define ASL_HEX_CONSTANT        3


/* File node - used for "Include" operator file stack */

typedef struct asl_file_node
{
    FILE                    *File;
    UINT32                  CurrentLineNumber;
```



```
    YY_BUFFER_STATE         State;
    char                    *Filename;
    struct asl_file_node    *Next;

} ASL_FILE_NODE;

/* File stack for the "Include" operator (NOT #include operator) */

ASL_FILE_NODE               *Gbl_IncludeFileStack = NULL;


/******************************************************************************
 *
 * FUNCTION:    AslDoLineDirective
 *
 * PARAMETERS:  None. Uses input() to access current source code line
 *
 * RETURN:      Updates global line number and filename
 *
 * DESCRIPTION: Handle #line directives emitted by the preprocessor.
 *
 * The #line directive is emitted by the preprocesser, and is used to
 * pass through line numbers from the original source code file to the
 * preprocessor output file (.i). This allows any compiler-generated
 * error messages to be displayed with the correct line number.
 *
 *****************************************************************************/

static void
AslDoLineDirective (
    void)
{
    int                     c;
    char                    *Token;
    UINT32                  LineNumber;
    char                    *Filename;
    UINT32                  i;


    /* Eat the entire line that contains the #line directive */

    Gbl_LineBufPtr = Gbl_CurrentLineBuffer;

    while ((c = input()) != '\n' && c != EOF)
    {
        *Gbl_LineBufPtr = c;
        Gbl_LineBufPtr++;
    }
    *Gbl_LineBufPtr = 0;

    /* First argument is the actual line number */

    Token = strtok (Gbl_CurrentLineBuffer, " ");
    if (!Token)
    {
        goto ResetAndExit;
    }

    /* First argument is the line number */

    LineNumber = (UINT32) UtDoConstant (Token);

    /* Emit the appropriate number of newlines */

    Gbl_CurrentColumn = 0;
    if (LineNumber > Gbl_CurrentLineNumber)
```

```
128     {
129         for (i = 0; i < (LineNumber - Gbl_CurrentLineNumber); i++)
130         {
131             FlWriteFile (ASL_FILE_SOURCE_OUTPUT, "\n", 1);
132             Gbl_CurrentColumn++;
133         }
134     }

136     FlSetLineNumber (LineNumber);

138     /* Second argument is the optional filename (in double quotes) */

140     Token = strtok (NULL, " \"");
141     if (Token)
142     {
143         Filename = ACPI_ALLOCATE_ZEROED (strlen (Token) + 1);
144         strcpy (Filename, Token);
145         FlSetFilename (Filename);
146     }

148     /* Third argument is not supported at this time */

150 ResetAndExit:

152     /* Reset globals for a new line */

154     Gbl_CurrentLineOffset += Gbl_CurrentColumn;
155     Gbl_CurrentColumn = 0;
156     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;
157 }


160 /*******************************************************************************
161  *
162  * FUNCTION:    AslPopInputFileStack
163  *
164  * PARAMETERS:  None
165  *
166  * RETURN:      0 if a node was popped, -1 otherwise
167  *
168  * DESCRIPTION: Pop the top of the input file stack and point the parser to
169  *              the saved parse buffer contained in the fnode. Also, set the
170  *              global line counters to the saved values. This function is
171  *              called when an include file reaches EOF.
172  *
173  ******************************************************************************/

175 int
176 AslPopInputFileStack (
177     void)
178 {
179     ASL_FILE_NODE           *Fnode;


182     Fnode = Gbl_IncludeFileStack;
183     DbgPrint (ASL_PARSE_OUTPUT, "\nPop InputFile Stack, Fnode %p\n\n", Fnode);

185     if (!Fnode)
186     {
187         return (-1);
188     }

190     /* Close the current include file */

192     fclose (yyin);
```



```
194     /* Update the top-of-stack */

196     Gbl_IncludeFileStack = Fnode->Next;

198     /* Reset global line counter and filename */

200     Gbl_Files[ASL_FILE_INPUT].Filename = Fnode->Filename;
201     Gbl_CurrentLineNumber = Fnode->CurrentLineNumber;

203     /* Point the parser to the popped file */

205     yy_delete_buffer (YY_CURRENT_BUFFER);
206     yy_switch_to_buffer (Fnode->State);

208     /* All done with this node */

210     ACPI_FREE (Fnode);
211     return (0);
212 }


215 /*******************************************************************************
216  *
217  * FUNCTION:    AslPushInputFileStack
218  *
219  * PARAMETERS:  InputFile           - Open file pointer
220  *              Filename            - Name of the file
221  *
222  * RETURN:      None
223  *
224  * DESCRIPTION: Push the InputFile onto the file stack, and point the parser
225  *              to this file. Called when an include file is successfully
226  *              opened.
227  *
228  ******************************************************************************/

230 void
231 AslPushInputFileStack (
232     FILE                    *InputFile,
233     char                    *Filename)
234 {
235     ASL_FILE_NODE           *Fnode;
236     YY_BUFFER_STATE         State;


239     /* Save the current state in an Fnode */

241     Fnode = UtLocalCalloc (sizeof (ASL_FILE_NODE));

243     Fnode->File                 = yyin;
244     Fnode->Next                 = Gbl_IncludeFileStack;
245     Fnode->State                = YY_CURRENT_BUFFER;
246     Fnode->CurrentLineNumber    = Gbl_CurrentLineNumber;
247     Fnode->Filename             = Gbl_Files[ASL_FILE_INPUT].Filename;

249     /* Push it on the stack */

251     Gbl_IncludeFileStack = Fnode;

253     /* Point the parser to this file */

255     State = yy_create_buffer (InputFile, YY_BUF_SIZE);
256     yy_switch_to_buffer (State);

258     DbgPrint (ASL_PARSE_OUTPUT, "\nPush InputFile Stack, returning %p\n\n", Inpu
```

```
260     /* Reset the global line count and filename */

262     Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
263     Gbl_CurrentLineNumber = 1;
264     yyin = InputFile;
265 }


268 /*******************************************************************************
269  *
270  * FUNCTION:    AslResetCurrentLineBuffer
271  *
272  * PARAMETERS:  None
273  *
274  * RETURN:      None
275  *
276  * DESCRIPTION: Reset the Line Buffer to zero, increment global line numbers.
277  *
278  ******************************************************************************/

280 void
281 AslResetCurrentLineBuffer (
282     void)
283 {

285     if (Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Handle)
286     {
287         FlWriteFile (ASL_FILE_SOURCE_OUTPUT, Gbl_CurrentLineBuffer,
288             Gbl_LineBufPtr - Gbl_CurrentLineBuffer);
289     }

291     Gbl_CurrentLineOffset += Gbl_CurrentColumn;
292     Gbl_CurrentColumn = 0;

294     Gbl_CurrentLineNumber++;
295     Gbl_LogicalLineNumber++;
296     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;
297 }


300 /*******************************************************************************
301  *
302  * FUNCTION:    AslInsertLineBuffer
303  *
304  * PARAMETERS:  SourceChar      - One char from the input ASL source file
305  *
306  * RETURN:      None
307  *
308  * DESCRIPTION: Put one character of the source file into the temp line buffer
309  *
310  ******************************************************************************/

312 void
313 AslInsertLineBuffer (
314     int                     SourceChar)
315 {
316     UINT32                  i;
317     UINT32                  Count = 1;


320     if (SourceChar == EOF)
321     {
322         return;
323     }

325     Gbl_InputByteCount++;
```



```
327     /* Handle tabs. Convert to spaces */

329     if (SourceChar == '\t')
330     {
331         SourceChar = ' ';
332         Count = ASL_SPACES_PER_TAB -
333                     (Gbl_CurrentColumn & (ASL_SPACES_PER_TAB-1));
334     }

336     for (i = 0; i < Count; i++)
337     {
338         Gbl_CurrentColumn++;

340         /* Insert the character into the line buffer */

342         *Gbl_LineBufPtr = (UINT8) SourceChar;
343         Gbl_LineBufPtr++;

345         if (Gbl_LineBufPtr > (Gbl_CurrentLineBuffer + (Gbl_LineBufferSize - 1)))
346         {
347 #if 0
348             /*
349              * Warning if we have split a long source line.
350              * <Probably overkill>
351              */
352             sprintf (MsgBuffer, "Max %u", Gbl_LineBufferSize);
353             AslCommonError (ASL_WARNING, ASL_MSG_LONG_LINE,
354                             Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
355                             Gbl_CurrentLineOffset, Gbl_CurrentColumn,
356                             Gbl_Files[ASL_FILE_INPUT].Filename, MsgBuffer);
357 #endif

359             AslResetCurrentLineBuffer ();
360         }
361         else if (SourceChar == '\n')
362         {
363             /* End of line */

365             AslResetCurrentLineBuffer ();
366         }
367     }
368 }


371 /*******************************************************************************
372  *
373  * FUNCTION:    count
374  *
375  * PARAMETERS:  yytext      - Contains the matched keyword.
376  *              Type        - Keyword/Character type:
377  *                            0 = anything except a keyword
378  *                            1 = pseudo-keywords
379  *                            2 = non-executable ASL keywords
380  *                            3 = executable ASL keywords
381  *
382  * RETURN:      None
383  *
384  * DESCRIPTION: Count keywords and put them into the line buffer
385  *
386  ******************************************************************************/

388 static void
389 count (
390     int                 Type)
391 {
```

```
392     int                 i;


395     switch (Type)
396     {
397     case 2:

399         TotalKeywords++;
400         TotalNamedObjects++;
401         break;

403     case 3:

405         TotalKeywords++;
406         TotalExecutableOpcodes++;
407         break;

409     default:

411         break;
412     }

414     for (i = 0; (yytext[i] != 0) && (yytext[i] != EOF); i++)
415     {
416         AslInsertLineBuffer (yytext[i]);
417         *Gbl_LineBufPtr = 0;
418     }
419 }


422 /******************************************************************************
423  *
424  * FUNCTION:    AslDoComment
425  *
426  * PARAMETERS:  none
427  *
428  * RETURN:      none
429  *
430  * DESCRIPTION: Process a standard comment.
431  *
432  *****************************************************************************/

434 static char
435 AslDoComment (
436     void)
437 {
438     int                 c;
439     int                 c1 = 0;


442     AslInsertLineBuffer ('/');
443     AslInsertLineBuffer ('*');

445 loop:

447     /* Eat chars until end-of-comment */

449     while ((c = input()) != '*' && c != EOF)
450     {
451         AslInsertLineBuffer (c);
452         c1 = c;
453     }

455     if (c == EOF)
456     {
457         goto EarlyEOF;
```



```
458     }

460     /*
461      * Check for nested comment -- can help catch cases where a previous
462      * comment was accidently left unterminated
463      */
464     if ((c1 == '/') && (c == '*'))
465     {
466         AslCommonError (ASL_WARNING, ASL_MSG_NESTED_COMMENT,
467                         Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
468                         Gbl_InputByteCount, Gbl_CurrentColumn,
469                         Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
470     }

472     /* Comment is closed only if the NEXT character is a slash */

474     AslInsertLineBuffer (c);

476     if ((c1 = input()) != '/' && c1 != EOF)
477     {
478         unput(c1);
479         goto loop;
480     }

482     if (c1 == EOF)
483     {
484         goto EarlyEOF;
485     }

487     AslInsertLineBuffer (c1);
488     return (TRUE);


491 EarlyEOF:
492     /*
493      * Premature End-Of-File
494      */
495     AslCommonError (ASL_ERROR, ASL_MSG_EARLY_EOF,
496                     Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
497                     Gbl_CurrentLineOffset, Gbl_CurrentColumn,
498                     Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
499     return (FALSE);
500 }


503 /******************************************************************************
504  *
505  * FUNCTION:    AslDoCommentType2
506  *
507  * PARAMETERS:  none
508  *
509  * RETURN:      none
510  *
511  * DESCRIPTION: Process a new "//" comment.
512  *
513  *****************************************************************************/

515 static char
516 AslDoCommentType2 (
517     void)
518 {
519     int                 c;


522     AslInsertLineBuffer ('/');
523     AslInsertLineBuffer ('/');
```

```
525     while ((c = input()) != '\n' && c != EOF)
526     {
527         AslInsertLineBuffer (c);
528     }
529
530     if (c == EOF)
531     {
532         /* End of file is OK, change to newline. Let parser detect EOF later */
533
534         c = '\n';
535     }
536
537     AslInsertLineBuffer (c);
538     return (TRUE);
539 }
540
541
542 /******************************************************************************
543  *
544  * FUNCTION:    AslDoStringLiteral
545  *
546  * PARAMETERS:  none
547  *
548  * RETURN:      none
549  *
550  * DESCRIPTION: Process a string literal (surrounded by quotes)
551  *
552  *****************************************************************************/
553
554 static char
555 AslDoStringLiteral (
556     void)
557 {
558     char                *StringBuffer = MsgBuffer;
559     char                *EndBuffer = MsgBuffer + ASL_MSG_BUFFER_SIZE;
560     char                *CleanString;
561     int                 StringChar;
562     UINT32              State = ASL_NORMAL_CHAR;
563     UINT32              i = 0;
564     UINT8               Digit;
565     char                ConvertBuffer[4];
566
567
568     /*
569      * Eat chars until end-of-literal.
570      * NOTE:  Put back the original surrounding quotes into the
571      * source line buffer.
572      */
573     AslInsertLineBuffer ('\"');
574     while ((StringChar = input()) != EOF)
575     {
576         AslInsertLineBuffer (StringChar);
577
578 DoCharacter:
579         switch (State)
580         {
581         case ASL_NORMAL_CHAR:
582
583             switch (StringChar)
584             {
585             case '\\':
586                 /*
587                  * Special handling for backslash-escape sequence. We will
588                  * toss the backslash and translate the escape char(s).
589                  */
```



```
590                 State = ASL_ESCAPE_SEQUENCE;
591                 continue;
592
593             case '\"':
594
595                 /* String terminator */
596
597                 goto CompletedString;
598
599             default:
600
601                 break;
602             }
603             break;
604
605
606         case ASL_ESCAPE_SEQUENCE:
607
608             State = ASL_NORMAL_CHAR;
609             switch (StringChar)
610             {
611             case 'a':
612
613                 StringChar = 0x07;      /* BELL */
614                 break;
615
616             case 'b':
617
618                 StringChar = 0x08;      /* BACKSPACE */
619                 break;
620
621             case 'f':
622
623                 StringChar = 0x0C;      /* FORMFEED */
624                 break;
625
626             case 'n':
627
628                 StringChar = 0x0A;      /* LINEFEED */
629                 break;
630
631             case 'r':
632
633                 StringChar = 0x0D;      /* CARRIAGE RETURN*/
634                 break;
635
636             case 't':
637
638                 StringChar = 0x09;      /* HORIZONTAL TAB */
639                 break;
640
641             case 'v':
642
643                 StringChar = 0x0B;      /* VERTICAL TAB */
644                 break;
645
646             case 'x':
647
648                 State = ASL_HEX_CONSTANT;
649                 i = 0;
650                 continue;
651
652             case '\'':                  /* Single Quote */
653             case '\"':                  /* Double Quote */
654             case '\\':                  /* Backslash */
```

```
656                 break;

658             default:

660                 /* Check for an octal digit (0-7) */

662                 if (ACPI_IS_OCTAL_DIGIT (StringChar))
663                 {
664                     State = ASL_OCTAL_CONSTANT;
665                     ConvertBuffer[0] = StringChar;
666                     i = 1;
667                     continue;
668                 }

670                 /* Unknown escape sequence issue warning, but use the character

672                 AslCommonError (ASL_WARNING, ASL_MSG_INVALID_ESCAPE,
673                                 Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
674                                 Gbl_CurrentLineOffset, Gbl_CurrentColumn,
675                                 Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
676                 break;
677             }
678             break;


681         case ASL_OCTAL_CONSTANT:

683             /* Up to three octal digits allowed */

685             if (!ACPI_IS_OCTAL_DIGIT (StringChar) ||
686                 (i > 2))
687             {
688                 /*
689                  * Reached end of the constant. Convert the assembled ASCII
690                  * string and resume processing of the next character
691                  */
692                 ConvertBuffer[i] = 0;
693                 Digit = (UINT8) ACPI_STRTOUL (ConvertBuffer, NULL, 8);

695                 /* Check for NULL or non-ascii character (ignore if so) */

697                 if ((Digit == 0) || (Digit > ACPI_ASCII_MAX))
698                 {
699                     AslCommonError (ASL_WARNING, ASL_MSG_INVALID_STRING,
700                                     Gbl_CurrentLineNumber, Gbl_LogicalLineNumber
701                                     Gbl_CurrentLineOffset, Gbl_CurrentColumn,
702                                     Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
703                 }
704                 else
705                 {
706                     *StringBuffer = (char) Digit;
707                     StringBuffer++;
708                     if (StringBuffer >= EndBuffer)
709                     {
710                         goto BufferOverflow;
711                     }
712                 }

714                 State = ASL_NORMAL_CHAR;
715                 goto DoCharacter;
716                 break;
717             }

719             /* Append another digit of the constant */

721             ConvertBuffer[i] = StringChar;
```



```
722             i++;
723             continue;

725         case ASL_HEX_CONSTANT:

727             /* Up to two hex digits allowed */

729             if (!ACPI_IS_XDIGIT (StringChar) ||
730                 (i > 1))
731             {
732                 /*
733                  * Reached end of the constant. Convert the assembled ASCII
734                  * string and resume processing of the next character
735                  */
736                 ConvertBuffer[i] = 0;
737                 Digit = (UINT8) ACPI_STRTOUL (ConvertBuffer, NULL, 16);

739                 /* Check for NULL or non-ascii character (ignore if so) */

741                 if ((Digit == 0) || (Digit > ACPI_ASCII_MAX))
742                 {
743                     AslCommonError (ASL_WARNING, ASL_MSG_INVALID_STRING,
744                                     Gbl_CurrentLineNumber, Gbl_LogicalLineNumber
745                                     Gbl_CurrentLineOffset, Gbl_CurrentColumn,
746                                     Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
747                 }
748                 else
749                 {
750                     *StringBuffer = (char) Digit;
751                     StringBuffer++;
752                     if (StringBuffer >= EndBuffer)
753                     {
754                         goto BufferOverflow;
755                     }
756                 }

758                 State = ASL_NORMAL_CHAR;
759                 goto DoCharacter;
760                 break;
761             }

763             /* Append another digit of the constant */

765             ConvertBuffer[i] = StringChar;
766             i++;
767             continue;

769         default:

771             break;
772         }

774         /* Save the finished character */

776         *StringBuffer = StringChar;
777         StringBuffer++;
778         if (StringBuffer >= EndBuffer)
779         {
780             goto BufferOverflow;
781         }
782     }

784     /*
785      * Premature End-Of-File
786      */
787     AslCommonError (ASL_ERROR, ASL_MSG_EARLY_EOF,
```

```
788                     Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
789                     Gbl_CurrentLineOffset, Gbl_CurrentColumn,
790                     Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
791     return (FALSE);


794 CompletedString:
795     /*
796      * Null terminate the input string and copy string to a new buffer
797      */
798     *StringBuffer = 0;

800     CleanString = UtGetStringBuffer (strlen (MsgBuffer) + 1);
801     if (!CleanString)
802     {
803         AslCommonError (ASL_ERROR, ASL_MSG_MEMORY_ALLOCATION,
804                         Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
805                         Gbl_CurrentLineOffset, Gbl_CurrentColumn,
806                         Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
807         return (FALSE);
808     }

810     ACPI_STRCPY (CleanString, MsgBuffer);
811     AslCompilerlval.s = CleanString;
812     return (TRUE);


815 BufferOverflow:

817     /* Literal was too long */

819     AslCommonError (ASL_ERROR, ASL_MSG_STRING_LENGTH,
820                     Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
821                     Gbl_CurrentLineOffset, Gbl_CurrentColumn,
822                     Gbl_Files[ASL_FILE_INPUT].Filename, "Max length 4096");
823     return (FALSE);
824 }
```

```
**********************************************************
   23962 Thu Dec 26 13:48:35 2013
new/usr/src/common/acpica/compiler/asltransform.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asltransform - Parse tree transforms
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asltransform")

/* Local prototypes */

static void
TrTransformSubtree (
    ACPI_PARSE_OBJECT       *Op);

static char *
TrAmlGetNextTempName (
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   *TempCount);
```



```
static void
TrAmlInitLineNumbers (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_PARSE_OBJECT       *Neighbor);

static void
TrAmlInitNode (
    ACPI_PARSE_OBJECT       *Op,
    UINT16                  ParseOpcode);

static void
TrAmlSetSubtreeParent (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_PARSE_OBJECT       *Parent);

static void
TrAmlInsertPeer (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_PARSE_OBJECT       *NewPeer);

static void
TrDoDefinitionBlock (
    ACPI_PARSE_OBJECT       *Op);

static void
TrDoSwitch (
    ACPI_PARSE_OBJECT       *StartNode);


/******************************************************************************
 *
 * FUNCTION:    TrAmlGetNextTempName
 *
 * PARAMETERS:  Op              - Current parse op
 *              TempCount       - Current temporary counter. Was originally
 *                                per-module; Currently per method, could be
 *                                expanded to per-scope.
 *
 * RETURN:      A pointer to name (allocated here).
 *
 * DESCRIPTION: Generate an ACPI name of the form _T_x. These names are
 *              reserved for use by the ASL compiler. (_T_0 through _T_Z)
 *
 *****************************************************************************/

static char *
TrAmlGetNextTempName (
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   *TempCount)
{
    char                    *TempName;


    if (*TempCount >= (10+26))  /* 0-35 valid: 0-9 and A-Z for TempName[3] */
    {
        /* Too many temps */

        AslError (ASL_ERROR, ASL_MSG_TOO_MANY_TEMPS, Op, NULL);
        return (NULL);
    }

    TempName = UtLocalCalloc (5);

    if (*TempCount < 10)    /* 0-9 */
    {
```

```
127         TempName[3] = (char) (*TempCount + '0');
128     }
129     else                    /* 10-35: A-Z */
130     {
131         TempName[3] = (char) (*TempCount + ('A' - 10));
132     }
133     (*TempCount)++;

135     /* First three characters are always "_T_" */

137     TempName[0] = '_';
138     TempName[1] = 'T';
139     TempName[2] = '_';

141     return (TempName);
142 }


145 /*******************************************************************************
146  *
147  * FUNCTION:    TrAmlInitLineNumbers
148  *
149  * PARAMETERS:  Op              - Op to be initialized
150  *              Neighbor        - Op used for initialization values
151  *
152  * RETURN:      None
153  *
154  * DESCRIPTION: Initialized the various line numbers for a parse node.
155  *
156  ******************************************************************************/

158 static void
159 TrAmlInitLineNumbers (
160     ACPI_PARSE_OBJECT       *Op,
161     ACPI_PARSE_OBJECT       *Neighbor)
162 {

164     Op->Asl.EndLine           = Neighbor->Asl.EndLine;
165     Op->Asl.EndLogicalLine    = Neighbor->Asl.EndLogicalLine;
166     Op->Asl.LineNumber        = Neighbor->Asl.LineNumber;
167     Op->Asl.LogicalByteOffset = Neighbor->Asl.LogicalByteOffset;
168     Op->Asl.LogicalLineNumber = Neighbor->Asl.LogicalLineNumber;
169 }


172 /*******************************************************************************
173  *
174  * FUNCTION:    TrAmlInitNode
175  *
176  * PARAMETERS:  Op              - Op to be initialized
177  *              ParseOpcode     - Opcode for this node
178  *
179  * RETURN:      None
180  *
181  * DESCRIPTION: Initialize a node with the parse opcode and opcode name.
182  *
183  ******************************************************************************/

185 static void
186 TrAmlInitNode (
187     ACPI_PARSE_OBJECT       *Op,
188     UINT16                  ParseOpcode)
189 {

191     Op->Asl.ParseOpcode = ParseOpcode;
192     UtSetParseOpName (Op);
```



```
193 }


196 /*******************************************************************************
197  *
198  * FUNCTION:    TrAmlSetSubtreeParent
199  *
200  * PARAMETERS:  Op              - First node in a list of peer nodes
201  *              Parent          - Parent of the subtree
202  *
203  * RETURN:      None
204  *
205  * DESCRIPTION: Set the parent for all peer nodes in a subtree
206  *
207  ******************************************************************************/

209 static void
210 TrAmlSetSubtreeParent (
211     ACPI_PARSE_OBJECT       *Op,
212     ACPI_PARSE_OBJECT       *Parent)
213 {
214     ACPI_PARSE_OBJECT       *Next;


217     Next = Op;
218     while (Next)
219     {
220         Next->Asl.Parent = Parent;
221         Next             = Next->Asl.Next;
222     }
223 }


226 /*******************************************************************************
227  *
228  * FUNCTION:    TrAmlInsertPeer
229  *
230  * PARAMETERS:  Op              - First node in a list of peer nodes
231  *              NewPeer         - Peer node to insert
232  *
233  * RETURN:      None
234  *
235  * DESCRIPTION: Insert a new peer node into a list of peers.
236  *
237  ******************************************************************************/

239 static void
240 TrAmlInsertPeer (
241     ACPI_PARSE_OBJECT       *Op,
242     ACPI_PARSE_OBJECT       *NewPeer)
243 {

245     NewPeer->Asl.Next = Op->Asl.Next;
246     Op->Asl.Next      = NewPeer;
247 }


250 /*******************************************************************************
251  *
252  * FUNCTION:    TrAmlTransformWalk
253  *
254  * PARAMETERS:  ASL_WALK_CALLBACK
255  *
256  * RETURN:      None
257  *
258  * DESCRIPTION: Parse tree walk to generate both the AML opcodes and the AML
```

```
 *              operands.
 *
 *****************************************************************************/

ACPI_STATUS
TrAmlTransformWalk (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{

    TrTransformSubtree (Op);
    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    TrTransformSubtree
 *
 * PARAMETERS:  Op        - The parent parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Prepare nodes to be output as AML data and operands. The more
 *              complex AML opcodes require processing of the child nodes
 *              (arguments/operands).
 *
 *****************************************************************************/

static void
TrTransformSubtree (
    ACPI_PARSE_OBJECT           *Op)
{

    if (Op->Asl.AmlOpcode == AML_RAW_DATA_BYTE)
    {
        return;
    }

    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_DEFINITIONBLOCK:

        TrDoDefinitionBlock (Op);
        break;

    case PARSEOP_SWITCH:

        TrDoSwitch (Op);
        break;

    case PARSEOP_METHOD:
        /*
         * TBD: Zero the tempname (_T_x) count. Probably shouldn't be a global,
         * however
         */
        Gbl_TempCount = 0;
        break;

    default:

        /* Nothing to do here for other opcodes */

        break;
    }
```



```
}


/*****************************************************************************
 *
 * FUNCTION:    TrDoDefinitionBlock
 *
 * PARAMETERS:  Op        - Parse node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Find the end of the definition block and set a global to this
 *              node. It is used by the compiler to insert compiler-generated
 *              names at the root level of the namespace.
 *
 *****************************************************************************/

static void
TrDoDefinitionBlock (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Next;
    UINT32                  i;


    Next = Op->Asl.Child;
    for (i = 0; i < 5; i++)
    {
        Next = Next->Asl.Next;
        if (i == 0)
        {
            /*
             * This is the table signature. Only the DSDT can be assumed
             * to be at the root of the namespace;  Therefore, namepath
             * optimization can only be performed on the DSDT.
             */
            if (!ACPI_COMPARE_NAME (Next->Asl.Value.String, ACPI_SIG_DSDT))
            {
                Gbl_ReferenceOptimizationFlag = FALSE;
            }
        }
    }

    Gbl_FirstLevelInsertionNode = Next;
}


/*****************************************************************************
 *
 * FUNCTION:    TrDoSwitch
 *
 * PARAMETERS:  StartNode        - Parse node for SWITCH
 *
 * RETURN:      None
 *
 *
 * DESCRIPTION: Translate ASL SWITCH statement to if/else pairs. There is
 *              no actual AML opcode for SWITCH -- it must be simulated.
 *
 *****************************************************************************/

static void
TrDoSwitch (
    ACPI_PARSE_OBJECT       *StartNode)
{
    ACPI_PARSE_OBJECT       *Next;
```

```
391     ACPI_PARSE_OBJECT       *CaseOp = NULL;
392     ACPI_PARSE_OBJECT       *CaseBlock = NULL;
393     ACPI_PARSE_OBJECT       *DefaultOp = NULL;
394     ACPI_PARSE_OBJECT       *CurrentParentNode;
395     ACPI_PARSE_OBJECT       *Conditional = NULL;
396     ACPI_PARSE_OBJECT       *Predicate;
397     ACPI_PARSE_OBJECT       *Peer;
398     ACPI_PARSE_OBJECT       *NewOp;
399     ACPI_PARSE_OBJECT       *NewOp2;
400     ACPI_PARSE_OBJECT       *MethodOp;
401     ACPI_PARSE_OBJECT       *StoreOp;
402     ACPI_PARSE_OBJECT       *BreakOp;
403     ACPI_PARSE_OBJECT       *BufferOp;
404     char                    *PredicateValueName;
405     UINT16                  Index;
406     UINT32                  Btype;


409     /* Start node is the Switch() node */

411     CurrentParentNode  = StartNode;

413     /* Create a new temp name of the form _T_x */

415     PredicateValueName = TrAmlGetNextTempName (StartNode, &Gbl_TempCount);
416     if (!PredicateValueName)
417     {
418         return;
419     }

421     /* First child is the Switch() predicate */

423     Next = StartNode->Asl.Child;

425     /*
426      * Examine the return type of the Switch Value -
427      * must be Integer/Buffer/String
428      */
429     Index = (UINT16) (Next->Asl.ParseOpcode - ASL_PARSE_OPCODE_BASE);
430     Btype = AslKeywordMapping[Index].AcpiBtype;
431     if ((Btype != ACPI_BTYPE_INTEGER) &&
432         (Btype != ACPI_BTYPE_STRING)  &&
433         (Btype != ACPI_BTYPE_BUFFER))
434     {
435         AslError (ASL_WARNING, ASL_MSG_SWITCH_TYPE, Next, NULL);
436         Btype = ACPI_BTYPE_INTEGER;
437     }

439     /* CASE statements start at next child */

441     Peer = Next->Asl.Next;
442     while (Peer)
443     {
444         Next = Peer;
445         Peer = Next->Asl.Next;

447         if (Next->Asl.ParseOpcode == PARSEOP_CASE)
448         {
449             if (CaseOp)
450             {
451                 /* Add an ELSE to complete the previous CASE */

453                 if (!Conditional)
454                 {
455                     return;
456                 }
```



```
457                 NewOp             = TrCreateLeafNode (PARSEOP_ELSE);
458                 NewOp->Asl.Parent = Conditional->Asl.Parent;
459                 TrAmlInitLineNumbers (NewOp, NewOp->Asl.Parent);

461                 /* Link ELSE node as a peer to the previous IF */

463                 TrAmlInsertPeer (Conditional, NewOp);
464                 CurrentParentNode = NewOp;
465             }

467             CaseOp      = Next;
468             Conditional = CaseOp;
469             CaseBlock   = CaseOp->Asl.Child->Asl.Next;
470             Conditional->Asl.Child->Asl.Next = NULL;
471             Predicate = CaseOp->Asl.Child;

473             if ((Predicate->Asl.ParseOpcode == PARSEOP_PACKAGE) ||
474                 (Predicate->Asl.ParseOpcode == PARSEOP_VAR_PACKAGE))
475             {
476                 /*
477                  * Convert the package declaration to this form:
478                  *
479                  * If (LNotEqual (Match (Package(<size>){<data>},
480                  *                       MEQ, _T_x, MTR, Zero, Zero), Ones))
481                  */
482                 NewOp2              = TrCreateLeafNode (PARSEOP_MATCHTYPE_MEQ);
483                 Predicate->Asl.Next = NewOp2;
484                 TrAmlInitLineNumbers (NewOp2, Conditional);

486                 NewOp               = NewOp2;
487                 NewOp2              = TrCreateValuedLeafNode (PARSEOP_NAMESTRING
488                                         (UINT64) ACPI_TO_INTEGER (PredicateValue
489                 NewOp->Asl.Next     = NewOp2;
490                 TrAmlInitLineNumbers (NewOp2, Predicate);

492                 NewOp               = NewOp2;
493                 NewOp2              = TrCreateLeafNode (PARSEOP_MATCHTYPE_MTR);
494                 NewOp->Asl.Next     = NewOp2;
495                 TrAmlInitLineNumbers (NewOp2, Predicate);

497                 NewOp               = NewOp2;
498                 NewOp2              = TrCreateLeafNode (PARSEOP_ZERO);
499                 NewOp->Asl.Next     = NewOp2;
500                 TrAmlInitLineNumbers (NewOp2, Predicate);

502                 NewOp               = NewOp2;
503                 NewOp2              = TrCreateLeafNode (PARSEOP_ZERO);
504                 NewOp->Asl.Next     = NewOp2;
505                 TrAmlInitLineNumbers (NewOp2, Predicate);

507                 NewOp2              = TrCreateLeafNode (PARSEOP_MATCH);
508                 NewOp2->Asl.Child   = Predicate;  /* PARSEOP_PACKAGE */
509                 TrAmlInitLineNumbers (NewOp2, Conditional);
510                 TrAmlSetSubtreeParent (Predicate, NewOp2);

512                 NewOp               = NewOp2;
513                 NewOp2              = TrCreateLeafNode (PARSEOP_ONES);
514                 NewOp->Asl.Next     = NewOp2;
515                 TrAmlInitLineNumbers (NewOp2, Conditional);

517                 NewOp2              = TrCreateLeafNode (PARSEOP_LEQUAL);
518                 NewOp2->Asl.Child   = NewOp;
519                 NewOp->Asl.Parent   = NewOp2;
520                 TrAmlInitLineNumbers (NewOp2, Conditional);
521                 TrAmlSetSubtreeParent (NewOp, NewOp2);
```

```
523                 NewOp               = NewOp2;
524                 NewOp2              = TrCreateLeafNode (PARSEOP_LNOT);
525                 NewOp2->Asl.Child   = NewOp;
526                 NewOp2->Asl.Parent  = Conditional;
527                 NewOp->Asl.Parent   = NewOp2;
528                 TrAmlInitLineNumbers (NewOp2, Conditional);

530                 Conditional->Asl.Child = NewOp2;
531                 NewOp2->Asl.Next = CaseBlock;
532             }
533             else
534             {
535                 /*
536                  * Integer and Buffer case.
537                  *
538                  * Change CaseOp() to:  If (LEqual (SwitchValue, CaseValue)) {..
539                  * Note: SwitchValue is first to allow the CaseValue to be impli
540                  * converted to the type of SwitchValue if necessary.
541                  *
542                  * CaseOp->Child is the case value
543                  * CaseOp->Child->Peer is the beginning of the case block
544                  */
545                 NewOp = TrCreateValuedLeafNode (PARSEOP_NAMESTRING,
546                             (UINT64) ACPI_TO_INTEGER (PredicateValueName));
547                 NewOp->Asl.Next = Predicate;
548                 TrAmlInitLineNumbers (NewOp, Predicate);

550                 NewOp2              = TrCreateLeafNode (PARSEOP_LEQUAL);
551                 NewOp2->Asl.Parent  = Conditional;
552                 NewOp2->Asl.Child   = NewOp;
553                 TrAmlInitLineNumbers (NewOp2, Conditional);

555                 TrAmlSetSubtreeParent (NewOp, NewOp2);

557                 Predicate           = NewOp2;
558                 Predicate->Asl.Next = CaseBlock;

560                 TrAmlSetSubtreeParent (Predicate, Conditional);
561                 Conditional->Asl.Child = Predicate;
562             }

564             /* Reinitialize the CASE node to an IF node */

566             TrAmlInitNode (Conditional, PARSEOP_IF);

568             /*
569              * The first CASE(IF) is not nested under an ELSE.
570              * All other CASEs are children of a parent ELSE.
571              */
572             if (CurrentParentNode == StartNode)
573             {
574                 Conditional->Asl.Next = NULL;
575             }
576             else
577             {
578                 /*
579                  * The IF is a child of previous IF/ELSE. It
580                  * is therefore without peer.
581                  */
582                 CurrentParentNode->Asl.Child = Conditional;
583                 Conditional->Asl.Parent      = CurrentParentNode;
584                 Conditional->Asl.Next        = NULL;
585             }
586         }
587         else if (Next->Asl.ParseOpcode == PARSEOP_DEFAULT)
588         {
```



```
589             if (DefaultOp)
590             {
591                 /*
592                  * More than one Default
593                  * (Parser does not catch this, must check here)
594                  */
595                 AslError (ASL_ERROR, ASL_MSG_MULTIPLE_DEFAULT, Next, NULL);
596             }
597             else
598             {
599                 /* Save the DEFAULT node for later, after CASEs */

601                 DefaultOp = Next;
602             }
603         }
604         else
605         {
606             /* Unknown peer opcode */

608             AcpiOsPrintf ("Unknown parse opcode for switch statement: %s (%u)\n"
609                         Next->Asl.ParseOpName, Next->Asl.ParseOpcode);
610         }
611     }

613     /* Add the default case at the end of the if/else construct */

615     if (DefaultOp)
616     {
617         /* If no CASE statements, this is an error - see below */

619         if (CaseOp)
620         {
621             /* Convert the DEFAULT node to an ELSE */

623             if (!Conditional)
624             {
625                 return;
626             }

628             TrAmlInitNode (DefaultOp, PARSEOP_ELSE);
629             DefaultOp->Asl.Parent = Conditional->Asl.Parent;

631             /* Link ELSE node as a peer to the previous IF */

633             TrAmlInsertPeer (Conditional, DefaultOp);
634         }
635     }

637     if (!CaseOp)
638     {
639         AslError (ASL_ERROR, ASL_MSG_NO_CASES, StartNode, NULL);
640     }


643     /*
644      * Create a Name(_T_x, ...) statement. This statement must appear at the
645      * method level, in case a loop surrounds the switch statement and could
646      * cause the name to be created twice (error).
647      */

649     /* Create the Name node */

651     Predicate = StartNode->Asl.Child;
652     NewOp = TrCreateLeafNode (PARSEOP_NAME);
653     TrAmlInitLineNumbers (NewOp, StartNode);
```

```
655     /* Find the parent method */

657     Next = StartNode;
658     while ((Next->Asl.ParseOpcode != PARSEOP_METHOD) &&
659            (Next->Asl.ParseOpcode != PARSEOP_DEFINITIONBLOCK))
660     {
661         Next = Next->Asl.Parent;
662     }
663     MethodOp = Next;

665     NewOp->Asl.CompileFlags |= NODE_COMPILER_EMITTED;
666     NewOp->Asl.Parent = Next;

668     /* Insert name after the method name and arguments */

670     Next = Next->Asl.Child; /* Name */
671     Next = Next->Asl.Next;  /* NumArgs */
672     Next = Next->Asl.Next;  /* SerializeRule */

674     /*
675      * If method is not Serialized, we must make is so, because of the way
676      * that Switch() must be implemented -- we cannot allow multiple threads
677      * to execute this method concurrently since we need to create local
678      * temporary name(s).
679      */
680     if (Next->Asl.ParseOpcode != PARSEOP_SERIALIZERULE_SERIAL)
681     {
682         AslError (ASL_REMARK, ASL_MSG_SERIALIZED, MethodOp, "Due to use of Switc
683         Next->Asl.ParseOpcode = PARSEOP_SERIALIZERULE_SERIAL;
684     }

686     Next = Next->Asl.Next;  /* SyncLevel */
687     Next = Next->Asl.Next;  /* ReturnType */
688     Next = Next->Asl.Next;  /* ParameterTypes */

690     TrAmlInsertPeer (Next, NewOp);
691     TrAmlInitLineNumbers (NewOp, Next);

693     /* Create the NameSeg child for the Name node */

695     NewOp2 = TrCreateValuedLeafNode (PARSEOP_NAMESEG,
696                 (UINT64) ACPI_TO_INTEGER (PredicateValueName));
697     TrAmlInitLineNumbers (NewOp2, NewOp);
698     NewOp2->Asl.CompileFlags |= NODE_IS_NAME_DECLARATION;
699     NewOp->Asl.Child  = NewOp2;

701     /* Create the initial value for the Name. Btype was already validated above

703     switch (Btype)
704     {
705     case ACPI_BTYPE_INTEGER:

707         NewOp2->Asl.Next = TrCreateValuedLeafNode (PARSEOP_ZERO,
708                                 (UINT64) 0);
709         TrAmlInitLineNumbers (NewOp2->Asl.Next, NewOp);
710         break;

712     case ACPI_BTYPE_STRING:

714         NewOp2->Asl.Next = TrCreateValuedLeafNode (PARSEOP_STRING_LITERAL,
715                                 (UINT64) ACPI_TO_INTEGER (""));
716         TrAmlInitLineNumbers (NewOp2->Asl.Next, NewOp);
717         break;

719     case ACPI_BTYPE_BUFFER:
```



```
721         (void) TrLinkPeerNode (NewOp2, TrCreateValuedLeafNode (PARSEOP_BUFFER,
722                                     (UINT64) 0));
723         Next = NewOp2->Asl.Next;
724         TrAmlInitLineNumbers (Next, NewOp2);
725         (void) TrLinkChildren (Next, 1, TrCreateValuedLeafNode (PARSEOP_ZERO,
726                                     (UINT64) 1));
727         TrAmlInitLineNumbers (Next->Asl.Child, Next);

729         BufferOp = TrCreateValuedLeafNode (PARSEOP_DEFAULT_ARG, (UINT64) 0);
730         TrAmlInitLineNumbers (BufferOp, Next->Asl.Child);
731         (void) TrLinkPeerNode (Next->Asl.Child, BufferOp);

733         TrAmlSetSubtreeParent (Next->Asl.Child, Next);
734         break;

736     default:

738         break;
739     }

741     TrAmlSetSubtreeParent (NewOp2, NewOp);

743     /*
744      * Transform the Switch() into a While(One)-Break node.
745      * And create a Store() node which will be used to save the
746      * Switch() value. The store is of the form: Store (Value, _T_x)
747      * where _T_x is the temp variable.
748      */
749     TrAmlInitNode (StartNode, PARSEOP_WHILE);
750     NewOp = TrCreateLeafNode (PARSEOP_ONE);
751     TrAmlInitLineNumbers (NewOp, StartNode);
752     NewOp->Asl.Next = Predicate->Asl.Next;
753     NewOp->Asl.Parent = StartNode;
754     StartNode->Asl.Child = NewOp;

756     /* Create a Store() node */

758     StoreOp = TrCreateLeafNode (PARSEOP_STORE);
759     TrAmlInitLineNumbers (StoreOp, NewOp);
760     StoreOp->Asl.Parent = StartNode;
761     TrAmlInsertPeer (NewOp, StoreOp);

763     /* Complete the Store subtree */

765     StoreOp->Asl.Child = Predicate;
766     Predicate->Asl.Parent = StoreOp;

768     NewOp = TrCreateValuedLeafNode (PARSEOP_NAMESEG,
769                 (UINT64) ACPI_TO_INTEGER (PredicateValueName));
770     TrAmlInitLineNumbers (NewOp, StoreOp);
771     NewOp->Asl.Parent    = StoreOp;
772     Predicate->Asl.Next  = NewOp;

774     /* Create a Break() node and insert it into the end of While() */

776     Conditional = StartNode->Asl.Child;
777     while (Conditional->Asl.Next)
778     {
779         Conditional = Conditional->Asl.Next;
780     }

782     BreakOp = TrCreateLeafNode (PARSEOP_BREAK);
783     TrAmlInitLineNumbers (BreakOp, NewOp);
784     BreakOp->Asl.Parent = StartNode;
785     TrAmlInsertPeer (Conditional, BreakOp);
786 }
```

```
**********************************************************
   32561 Thu Dec 26 13:48:35 2013
new/usr/src/common/acpica/compiler/asltree.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asltree - parse tree management
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acapps.h"
#include <time.h>

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asltree")

/* Local prototypes */

static ACPI_PARSE_OBJECT *
TrGetNextNode (
    void);

static char *
TrGetNodeFlagName (
```



```
    UINT32                  Flags);


/******************************************************************************
 *
 * FUNCTION:    TrGetNextNode
 *
 * PARAMETERS:  None
 *
 * RETURN:      New parse node. Aborts on allocation failure
 *
 * DESCRIPTION: Allocate a new parse node for the parse tree. Bypass the local
 *              dynamic memory manager for performance reasons (This has a
 *              major impact on the speed of the compiler.)
 *
 *****************************************************************************/

static ACPI_PARSE_OBJECT *
TrGetNextNode (
    void)
{

    if (Gbl_NodeCacheNext >= Gbl_NodeCacheLast)
    {
        Gbl_NodeCacheNext = UtLocalCalloc (sizeof (ACPI_PARSE_OBJECT) *
                                ASL_NODE_CACHE_SIZE);
        Gbl_NodeCacheLast = Gbl_NodeCacheNext + ASL_NODE_CACHE_SIZE;
    }

    return (Gbl_NodeCacheNext++);
}


/******************************************************************************
 *
 * FUNCTION:    TrAllocateNode
 *
 * PARAMETERS:  ParseOpcode         - Opcode to be assigned to the node
 *
 * RETURN:      New parse node. Aborts on allocation failure
 *
 * DESCRIPTION: Allocate and initialize a new parse node for the parse tree
 *
 *****************************************************************************/

ACPI_PARSE_OBJECT *
TrAllocateNode (
    UINT32                  ParseOpcode)
{
    ACPI_PARSE_OBJECT       *Op;


    Op = TrGetNextNode ();

    Op->Asl.ParseOpcode       = (UINT16) ParseOpcode;
    Op->Asl.Filename          = Gbl_Files[ASL_FILE_INPUT].Filename;
    Op->Asl.LineNumber        = Gbl_CurrentLineNumber;
    Op->Asl.LogicalLineNumber = Gbl_LogicalLineNumber;
    Op->Asl.LogicalByteOffset = Gbl_CurrentLineOffset;
    Op->Asl.Column            = Gbl_CurrentColumn;

    UtSetParseOpName (Op);
    return (Op);
}
```

```
/*******************************************************************************
 *
 * FUNCTION:    TrReleaseNode
 *
 * PARAMETERS:  Op            - Op to be released
 *
 * RETURN:      None
 *
 * DESCRIPTION: "release" a node. In truth, nothing is done since the node
 *              is part of a larger buffer
 *
 ******************************************************************************/

void
TrReleaseNode (
    ACPI_PARSE_OBJECT       *Op)
{

    return;
}


/*******************************************************************************
 *
 * FUNCTION:    TrUpdateNode
 *
 * PARAMETERS:  ParseOpcode         - New opcode to be assigned to the node
 *              Op                - An existing parse node
 *
 * RETURN:      The updated node
 *
 * DESCRIPTION: Change the parse opcode assigned to a node. Usually used to
 *              change an opcode to DEFAULT_ARG so that the node is ignored
 *              during the code generation. Also used to set generic integers
 *              to a specific size (8, 16, 32, or 64 bits)
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
TrUpdateNode (
    UINT32                  ParseOpcode,
    ACPI_PARSE_OBJECT       *Op)
{

    if (!Op)
    {
        return (NULL);
    }

    DbgPrint (ASL_PARSE_OUTPUT,
        "\nUpdateNode: Old - %s, New - %s\n\n",
        UtGetOpName (Op->Asl.ParseOpcode),
        UtGetOpName (ParseOpcode));

    /* Assign new opcode and name */

    if (Op->Asl.ParseOpcode == PARSEOP_ONES)
    {
        switch (ParseOpcode)
        {
        case PARSEOP_BYTECONST:

            Op->Asl.Value.Integer = ACPI_UINT8_MAX;
            break;

        case PARSEOP_WORDCONST:
```



```
            Op->Asl.Value.Integer = ACPI_UINT16_MAX;
            break;

        case PARSEOP_DWORDCONST:

            Op->Asl.Value.Integer = ACPI_UINT32_MAX;
            break;

        /* Don't need to do the QWORD case */

        default:

            /* Don't care about others */
            break;
        }
    }

    Op->Asl.ParseOpcode = (UINT16) ParseOpcode;
    UtSetParseOpName (Op);

    /*
     * For the BYTE, WORD, and DWORD constants, make sure that the integer
     * that was passed in will actually fit into the data type
     */
    switch (ParseOpcode)
    {
    case PARSEOP_BYTECONST:

        UtCheckIntegerRange (Op, 0x00, ACPI_UINT8_MAX);
        Op->Asl.Value.Integer &= ACPI_UINT8_MAX;
        break;

    case PARSEOP_WORDCONST:

        UtCheckIntegerRange (Op, 0x00, ACPI_UINT16_MAX);
        Op->Asl.Value.Integer &= ACPI_UINT16_MAX;
        break;

    case PARSEOP_DWORDCONST:

        UtCheckIntegerRange (Op, 0x00, ACPI_UINT32_MAX);
        Op->Asl.Value.Integer &= ACPI_UINT32_MAX;
        break;

    default:

        /* Don't care about others, don't need to check QWORD */

        break;
    }

    return (Op);
}


/*******************************************************************************
 *
 * FUNCTION:    TrGetNodeFlagName
 *
 * PARAMETERS:  Flags               - Flags word to be decoded
 *
 * RETURN:      Name string. Always returns a valid string pointer.
 *
 * DESCRIPTION: Decode a flags word
 *
```

```
 *****************************************************************************/

static char *
TrGetNodeFlagName (
    UINT32                  Flags)
{

    switch (Flags)
    {
    case NODE_VISITED:

        return ("NODE_VISITED");

    case NODE_AML_PACKAGE:

        return ("NODE_AML_PACKAGE");

    case NODE_IS_TARGET:

        return ("NODE_IS_TARGET");

    case NODE_IS_RESOURCE_DESC:

        return ("NODE_IS_RESOURCE_DESC");

    case NODE_IS_RESOURCE_FIELD:

        return ("NODE_IS_RESOURCE_FIELD");

    case NODE_HAS_NO_EXIT:

        return ("NODE_HAS_NO_EXIT");

    case NODE_IF_HAS_NO_EXIT:

        return ("NODE_IF_HAS_NO_EXIT");

    case NODE_NAME_INTERNALIZED:

        return ("NODE_NAME_INTERNALIZED");

    case NODE_METHOD_NO_RETVAL:

        return ("NODE_METHOD_NO_RETVAL");

    case NODE_METHOD_SOME_NO_RETVAL:

        return ("NODE_METHOD_SOME_NO_RETVAL");

    case NODE_RESULT_NOT_USED:

        return ("NODE_RESULT_NOT_USED");

    case NODE_METHOD_TYPED:

        return ("NODE_METHOD_TYPED");

    case NODE_COMPILE_TIME_CONST:

        return ("NODE_COMPILE_TIME_CONST");

    case NODE_IS_TERM_ARG:

        return ("NODE_IS_TERM_ARG");

    case NODE_WAS_ONES_OP:
```



```
        return ("NODE_WAS_ONES_OP");

    case NODE_IS_NAME_DECLARATION:

        return ("NODE_IS_NAME_DECLARATION");

    default:

        return ("Multiple Flags (or unknown flag) set");
    }
}


/*******************************************************************************
 *
 * FUNCTION:    TrSetNodeFlags
 *
 * PARAMETERS:  Op                  - An existing parse node
 *              Flags               - New flags word
 *
 * RETURN:      The updated parser op
 *
 * DESCRIPTION: Set bits in the node flags word. Will not clear bits, only set
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
TrSetNodeFlags (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Flags)
{

    DbgPrint (ASL_PARSE_OUTPUT,
        "\nSetNodeFlags: Op %p, %8.8X %s\n\n", Op, Flags,
        TrGetNodeFlagName (Flags));

    if (!Op)
    {
        return (NULL);
    }

    Op->Asl.CompileFlags |= Flags;
    return (Op);
}


/*******************************************************************************
 *
 * FUNCTION:    TrSetNodeAmlLength
 *
 * PARAMETERS:  Op                  - An existing parse node
 *              Length              - AML Length
 *
 * RETURN:      The updated parser op
 *
 * DESCRIPTION: Set the AML Length in a node. Used by the parser to indicate
 *              the presence of a node that must be reduced to a fixed length
 *              constant.
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
TrSetNodeAmlLength (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Length)
```

```
{

    DbgPrint (ASL_PARSE_OUTPUT,
        "\nSetNodeAmlLength: Op %p, %8.8X\n", Op, Length);

    if (!Op)
    {
        return (NULL);
    }

    Op->Asl.AmlLength = Length;
    return (Op);
}


/*******************************************************************************
 *
 * FUNCTION:    TrSetEndLineNumber
 *
 * PARAMETERS:  Op                - An existing parse node
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Set the ending line numbers (file line and logical line) of a
 *              parse node to the current line numbers.
 *
 ******************************************************************************/

void
TrSetEndLineNumber (
    ACPI_PARSE_OBJECT       *Op)
{

    /* If the end line # is already set, just return */

    if (Op->Asl.EndLine)
    {
        return;
    }

    Op->Asl.EndLine        = Gbl_CurrentLineNumber;
    Op->Asl.EndLogicalLine = Gbl_LogicalLineNumber;
}


/*******************************************************************************
 *
 * FUNCTION:    TrCreateLeafNode
 *
 * PARAMETERS:  ParseOpcode         - New opcode to be assigned to the node
 *
 * RETURN:      Pointer to the new node. Aborts on allocation failure
 *
 * DESCRIPTION: Create a simple leaf node (no children or peers, and no value
 *              assigned to the node)
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
TrCreateLeafNode (
    UINT32                  ParseOpcode)
{
    ACPI_PARSE_OBJECT       *Op;


    Op = TrAllocateNode (ParseOpcode);
```



```
    DbgPrint (ASL_PARSE_OUTPUT,
        "\nCreateLeafNode  Ln/Col %u/%u NewNode %p  Op %s\n\n",
        Op->Asl.LineNumber, Op->Asl.Column, Op, UtGetOpName(ParseOpcode));

    return (Op);
}


/*******************************************************************************
 *
 * FUNCTION:    TrCreateConstantLeafNode
 *
 * PARAMETERS:  ParseOpcode         - The constant opcode
 *
 * RETURN:      Pointer to the new node. Aborts on allocation failure
 *
 * DESCRIPTION: Create a leaf node (no children or peers) for one of the
 *              special constants - __LINE__, __FILE__, and __DATE__.
 *
 * Note: An implemenation of __FUNC__ cannot happen here because we don't
 * have a full parse tree at this time and cannot find the parent control
 * method. If it is ever needed, __FUNC__ must be implemented later, after
 * the parse tree has been fully constructed.
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
TrCreateConstantLeafNode (
    UINT32                  ParseOpcode)
{
    ACPI_PARSE_OBJECT       *Op = NULL;
    time_t                  CurrentTime;
    char                    *StaticTimeString;
    char                    *TimeString;
    char                    *Path;
    char                    *Filename;


    switch (ParseOpcode)
    {
    case PARSEOP___LINE__:

        Op = TrAllocateNode (PARSEOP_INTEGER);
        Op->Asl.Value.Integer = Op->Asl.LineNumber;
        break;

    case PARSEOP___PATH__:

        Op = TrAllocateNode (PARSEOP_STRING_LITERAL);

        /* Op.Asl.Filename contains the full pathname to the file */

        Op->Asl.Value.String = Op->Asl.Filename;
        break;

    case PARSEOP___FILE__:

        Op = TrAllocateNode (PARSEOP_STRING_LITERAL);

        /* Get the simple filename from the full path */

        FlSplitInputPathname (Op->Asl.Filename, &Path, &Filename);
        ACPI_FREE (Path);
        Op->Asl.Value.String = Filename;
        break;
```

```
524     case PARSEOP___DATE__:

526         Op = TrAllocateNode (PARSEOP_STRING_LITERAL);

528         /* Get a copy of the current time */

530         CurrentTime = time (NULL);
531         StaticTimeString = ctime (&CurrentTime);
532         TimeString = UtLocalCalloc (strlen (StaticTimeString) + 1);
533         strcpy (TimeString, StaticTimeString);

535         TimeString[strlen(TimeString) -1] = 0;  /* Remove trailing newline */
536         Op->Asl.Value.String = TimeString;
537         break;

539     default: /* This would be an internal error */

541         return (NULL);
542     }

544     DbgPrint (ASL_PARSE_OUTPUT,
545         "\nCreateConstantLeafNode  Ln/Col %u/%u NewNode %p  Op %s  Value %8.8X%8
546         Op->Asl.LineNumber, Op->Asl.Column, Op, UtGetOpName (ParseOpcode),
547         ACPI_FORMAT_UINT64 (Op->Asl.Value.Integer));
548     return (Op);
549 }


552 /*******************************************************************************
553  *
554  * FUNCTION:    TrCreateValuedLeafNode
555  *
556  * PARAMETERS:  ParseOpcode         - New opcode to be assigned to the node
557  *              Value               - Value to be assigned to the node
558  *
559  * RETURN:      Pointer to the new node. Aborts on allocation failure
560  *
561  * DESCRIPTION: Create a leaf node (no children or peers) with a value
562  *              assigned to it
563  *
564  ******************************************************************************/

566 ACPI_PARSE_OBJECT *
567 TrCreateValuedLeafNode (
568     UINT32                  ParseOpcode,
569     UINT64                  Value)
570 {
571     ACPI_PARSE_OBJECT       *Op;


574     Op = TrAllocateNode (ParseOpcode);

576     DbgPrint (ASL_PARSE_OUTPUT,
577         "\nCreateValuedLeafNode  Ln/Col %u/%u NewNode %p  Op %s  Value %8.8X%8.8
578         Op->Asl.LineNumber, Op->Asl.Column, Op, UtGetOpName(ParseOpcode),
579         ACPI_FORMAT_UINT64 (Value));
580     Op->Asl.Value.Integer = Value;

582     switch (ParseOpcode)
583     {
584     case PARSEOP_STRING_LITERAL:

586         DbgPrint (ASL_PARSE_OUTPUT, "STRING->%s", Value);
587         break;
```



```
589     case PARSEOP_NAMESEG:

591         DbgPrint (ASL_PARSE_OUTPUT, "NAMESEG->%s", Value);
592         break;

594     case PARSEOP_NAMESTRING:

596         DbgPrint (ASL_PARSE_OUTPUT, "NAMESTRING->%s", Value);
597         break;

599     case PARSEOP_EISAID:

601         DbgPrint (ASL_PARSE_OUTPUT, "EISAID->%s", Value);
602         break;

604     case PARSEOP_METHOD:

606         DbgPrint (ASL_PARSE_OUTPUT, "METHOD");
607         break;

609     case PARSEOP_INTEGER:

611         DbgPrint (ASL_PARSE_OUTPUT, "INTEGER");
612         break;

614     default:

616         break;
617     }

619     DbgPrint (ASL_PARSE_OUTPUT, "\n\n");
620     return (Op);
621 }


624 /*******************************************************************************
625  *
626  * FUNCTION:    TrCreateNode
627  *
628  * PARAMETERS:  ParseOpcode         - Opcode to be assigned to the node
629  *              NumChildren         - Number of children to follow
630  *              ...                 - A list of child nodes to link to the new
631  *                                    node. NumChildren long.
632  *
633  * RETURN:      Pointer to the new node. Aborts on allocation failure
634  *
635  * DESCRIPTION: Create a new parse node and link together a list of child
636  *              nodes underneath the new node.
637  *
638  ******************************************************************************/

640 ACPI_PARSE_OBJECT *
641 TrCreateNode (
642     UINT32                  ParseOpcode,
643     UINT32                  NumChildren,
644     ...)
645 {
646     ACPI_PARSE_OBJECT       *Op;
647     ACPI_PARSE_OBJECT       *Child;
648     ACPI_PARSE_OBJECT       *PrevChild;
649     va_list                 ap;
650     UINT32                  i;
651     BOOLEAN                 FirstChild;


654     va_start (ap, NumChildren);
```

```
656     /* Allocate one new node */

658     Op = TrAllocateNode (ParseOpcode);

660     DbgPrint (ASL_PARSE_OUTPUT,
661         "\nCreateNode  Ln/Col %u/%u NewParent %p Child %u Op %s  ",
662         Op->Asl.LineNumber, Op->Asl.Column, Op, NumChildren, UtGetOpName(ParseOp

664     /* Some extra debug output based on the parse opcode */

666     switch (ParseOpcode)
667     {
668     case PARSEOP_DEFINITIONBLOCK:

670         RootNode = Op;
671         DbgPrint (ASL_PARSE_OUTPUT, "DEFINITION_BLOCK (Tree Completed)->");
672         break;

674     case PARSEOP_OPERATIONREGION:

676         DbgPrint (ASL_PARSE_OUTPUT, "OPREGION->");
677         break;

679     case PARSEOP_OR:

681         DbgPrint (ASL_PARSE_OUTPUT, "OR->");
682         break;

684     default:

686         /* Nothing to do for other opcodes */

688         break;
689     }

691     /* Link the new node to its children */

693     PrevChild = NULL;
694     FirstChild = TRUE;
695     for (i = 0; i < NumChildren; i++)
696     {
697         /* Get the next child */

699         Child = va_arg (ap, ACPI_PARSE_OBJECT *);
700         DbgPrint (ASL_PARSE_OUTPUT, "%p, ", Child);

702         /*
703          * If child is NULL, this means that an optional argument
704          * was omitted. We must create a placeholder with a special
705          * opcode (DEFAULT_ARG) so that the code generator will know
706          * that it must emit the correct default for this argument
707          */
708         if (!Child)
709         {
710             Child = TrAllocateNode (PARSEOP_DEFAULT_ARG);
711         }

713         /* Link first child to parent */

715         if (FirstChild)
716         {
717             FirstChild = FALSE;
718             Op->Asl.Child = Child;
719         }
```



```
721         /* Point all children to parent */

723         Child->Asl.Parent = Op;

725         /* Link children in a peer list */

727         if (PrevChild)
728         {
729             PrevChild->Asl.Next = Child;
730         };

732         /*
733          * This child might be a list, point all nodes in the list
734          * to the same parent
735          */
736         while (Child->Asl.Next)
737         {
738             Child = Child->Asl.Next;
739             Child->Asl.Parent = Op;
740         }

742         PrevChild = Child;
743     }
744     va_end(ap);

746     DbgPrint (ASL_PARSE_OUTPUT, "\n\n");
747     return (Op);
748 }


751 /*******************************************************************************
752  *
753  * FUNCTION:    TrLinkChildren
754  *
755  * PARAMETERS:  Op                - An existing parse node
756  *              NumChildren         - Number of children to follow
757  *              ...                 - A list of child nodes to link to the new
758  *                                    node. NumChildren long.
759  *
760  * RETURN:      The updated (linked) node
761  *
762  * DESCRIPTION: Link a group of nodes to an existing parse node
763  *
764  ******************************************************************************/

766 ACPI_PARSE_OBJECT *
767 TrLinkChildren (
768     ACPI_PARSE_OBJECT       *Op,
769     UINT32                  NumChildren,
770     ...)
771 {
772     ACPI_PARSE_OBJECT       *Child;
773     ACPI_PARSE_OBJECT       *PrevChild;
774     va_list                 ap;
775     UINT32                  i;
776     BOOLEAN                 FirstChild;


779     va_start (ap, NumChildren);


782     TrSetEndLineNumber (Op);

784     DbgPrint (ASL_PARSE_OUTPUT,
785         "\nLinkChildren  Line [%u to %u] NewParent %p Child %u Op %s  ",
786         Op->Asl.LineNumber, Op->Asl.EndLine,
```

```
787         Op, NumChildren, UtGetOpName(Op->Asl.ParseOpcode));

789     switch (Op->Asl.ParseOpcode)
790     {
791     case PARSEOP_DEFINITIONBLOCK:

793         RootNode = Op;
794         DbgPrint (ASL_PARSE_OUTPUT, "DEFINITION_BLOCK (Tree Completed)->");
795         break;

797     case PARSEOP_OPERATIONREGION:

799         DbgPrint (ASL_PARSE_OUTPUT, "OPREGION->");
800         break;

802     case PARSEOP_OR:

804         DbgPrint (ASL_PARSE_OUTPUT, "OR->");
805         break;

807     default:

809         /* Nothing to do for other opcodes */

811         break;
812     }

814     /* Link the new node to it's children */

816     PrevChild = NULL;
817     FirstChild = TRUE;
818     for (i = 0; i < NumChildren; i++)
819     {
820         Child = va_arg (ap, ACPI_PARSE_OBJECT *);

822         if ((Child == PrevChild) && (Child != NULL))
823         {
824             AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Child,
825                 "Child node list invalid");
826             va_end(ap);
827             return (Op);
828         }

830         DbgPrint (ASL_PARSE_OUTPUT, "%p, ", Child);

832         /*
833          * If child is NULL, this means that an optional argument
834          * was omitted. We must create a placeholder with a special
835          * opcode (DEFAULT_ARG) so that the code generator will know
836          * that it must emit the correct default for this argument
837          */
838         if (!Child)
839         {
840             Child = TrAllocateNode (PARSEOP_DEFAULT_ARG);
841         }

843         /* Link first child to parent */

845         if (FirstChild)
846         {
847             FirstChild = FALSE;
848             Op->Asl.Child = Child;
849         }

851         /* Point all children to parent */
```



```
853         Child->Asl.Parent = Op;

855         /* Link children in a peer list */

857         if (PrevChild)
858         {
859             PrevChild->Asl.Next = Child;
860         };

862         /*
863          * This child might be a list, point all nodes in the list
864          * to the same parent
865          */
866         while (Child->Asl.Next)
867         {
868             Child = Child->Asl.Next;
869             Child->Asl.Parent = Op;
870         }
871         PrevChild = Child;
872     }

874     va_end(ap);
875     DbgPrint (ASL_PARSE_OUTPUT, "\n\n");
876     return (Op);
877 }


880 /*******************************************************************************
881  *
882  * FUNCTION:    TrLinkPeerNode
883  *
884  * PARAMETERS:  Op1           - First peer
885  *              Op2           - Second peer
886  *
887  * RETURN:      Op1 or the non-null node.
888  *
889  * DESCRIPTION: Link two nodes as peers. Handles cases where one peer is null.
890  *
891  ******************************************************************************/

893 ACPI_PARSE_OBJECT *
894 TrLinkPeerNode (
895     ACPI_PARSE_OBJECT       *Op1,
896     ACPI_PARSE_OBJECT       *Op2)
897 {
898     ACPI_PARSE_OBJECT       *Next;


901     DbgPrint (ASL_PARSE_OUTPUT,
902         "\nLinkPeerNode: 1=%p (%s), 2=%p (%s)\n\n",
903         Op1, Op1 ? UtGetOpName(Op1->Asl.ParseOpcode) : NULL,
904         Op2, Op2 ? UtGetOpName(Op2->Asl.ParseOpcode) : NULL);


907     if ((!Op1) && (!Op2))
908     {
909         DbgPrint (ASL_PARSE_OUTPUT, "\nTwo Null nodes!\n");
910         return (Op1);
911     }

913     /* If one of the nodes is null, just return the non-null node */

915     if (!Op2)
916     {
917         return (Op1);
918     }
```

```
920     if (!Op1)
921     {
922         return (Op2);
923     }

925     if (Op1 == Op2)
926     {
927         DbgPrint (ASL_DEBUG_OUTPUT,
928             "\n\n************* Internal error, linking node to itself %p\n\n\n",
929             Op1);
930         AslError (ASL_WARNING, ASL_MSG_COMPILER_INTERNAL, Op1,
931             "Linking node to itself");
932         return (Op1);
933     }

935     Op1->Asl.Parent = Op2->Asl.Parent;

937     /*
938      * Op 1 may already have a peer list (such as an IF/ELSE pair),
939      * so we must walk to the end of the list and attach the new
940      * peer at the end
941      */
942     Next = Op1;
943     while (Next->Asl.Next)
944     {
945         Next = Next->Asl.Next;
946     }

948     Next->Asl.Next = Op2;
949     return (Op1);
950 }


953 /*******************************************************************************
954  *
955  * FUNCTION:    TrLinkPeerNodes
956  *
957  * PARAMETERS:  NumPeers            - The number of nodes in the list to follow
958  *              ...                 - A list of nodes to link together as peers
959  *
960  * RETURN:      The first node in the list (head of the peer list)
961  *
962  * DESCRIPTION: Link together an arbitrary number of peer nodes.
963  *
964  ******************************************************************************/

966 ACPI_PARSE_OBJECT *
967 TrLinkPeerNodes (
968     UINT32                  NumPeers,
969     ...)
970 {
971     ACPI_PARSE_OBJECT       *This;
972     ACPI_PARSE_OBJECT       *Next;
973     va_list                 ap;
974     UINT32                  i;
975     ACPI_PARSE_OBJECT       *Start;


978     DbgPrint (ASL_PARSE_OUTPUT,
979         "\nLinkPeerNodes: (%u) ", NumPeers);

981     va_start (ap, NumPeers);
982     This = va_arg (ap, ACPI_PARSE_OBJECT *);
983     Start = This;
```



```
985     /*
986      * Link all peers
987      */
988     for (i = 0; i < (NumPeers -1); i++)
989     {
990         DbgPrint (ASL_PARSE_OUTPUT, "%u=%p ", (i+1), This);

992         while (This->Asl.Next)
993         {
994             This = This->Asl.Next;
995         }

997         /* Get another peer node */

999         Next = va_arg (ap, ACPI_PARSE_OBJECT *);
1000        if (!Next)
1001        {
1002            Next = TrAllocateNode (PARSEOP_DEFAULT_ARG);
1003        }

1005        /* link new node to the current node */

1007        This->Asl.Next = Next;
1008        This = Next;
1009    }
1010    va_end (ap);

1012    DbgPrint (ASL_PARSE_OUTPUT,"\n\n");
1013    return (Start);
1014 }


1017 /*******************************************************************************
1018  *
1019  * FUNCTION:    TrLinkChildNode
1020  *
1021  * PARAMETERS:  Op1             - Parent node
1022  *              Op2             - Op to become a child
1023  *
1024  * RETURN:      The parent node
1025  *
1026  * DESCRIPTION: Link two nodes together as a parent and child
1027  *
1028  ******************************************************************************/

1030 ACPI_PARSE_OBJECT *
1031 TrLinkChildNode (
1032     ACPI_PARSE_OBJECT       *Op1,
1033     ACPI_PARSE_OBJECT       *Op2)
1034 {
1035     ACPI_PARSE_OBJECT       *Next;


1038    DbgPrint (ASL_PARSE_OUTPUT,
1039        "\nLinkChildNode: Parent=%p (%s), Child=%p (%s)\n\n",
1040        Op1, Op1 ? UtGetOpName(Op1->Asl.ParseOpcode): NULL,
1041        Op2, Op2 ? UtGetOpName(Op2->Asl.ParseOpcode): NULL);

1043    if (!Op1 || !Op2)
1044    {
1045        return (Op1);
1046    }

1048    Op1->Asl.Child = Op2;

1050    /* Set the child and all peers of the child to point to the parent */
```

```
1052     Next = Op2;
1053     while (Next)
1054     {
1055         Next->Asl.Parent = Op1;
1056         Next = Next->Asl.Next;
1057     }

1059     return (Op1);
1060 }


1063 /*******************************************************************************
1064  *
1065  * FUNCTION:    TrWalkParseTree
1066  *
1067  * PARAMETERS:  Visitation              - Type of walk
1068  *              DescendingCallback      - Called during tree descent
1069  *              AscendingCallback       - Called during tree ascent
1070  *              Context                 - To be passed to the callbacks
1071  *
1072  * RETURN:      Status from callback(s)
1073  *
1074  * DESCRIPTION: Walk the entire parse tree.
1075  *
1076  ******************************************************************************/

1078 ACPI_STATUS
1079 TrWalkParseTree (
1080     ACPI_PARSE_OBJECT       *Op,
1081     UINT32                  Visitation,
1082     ASL_WALK_CALLBACK       DescendingCallback,
1083     ASL_WALK_CALLBACK       AscendingCallback,
1084     void                    *Context)
1085 {
1086     UINT32                  Level;
1087     BOOLEAN                 NodePreviouslyVisited;
1088     ACPI_PARSE_OBJECT       *StartOp = Op;
1089     ACPI_STATUS             Status;


1092     if (!RootNode)
1093     {
1094         return (AE_OK);
1095     }

1097     Level = 0;
1098     NodePreviouslyVisited = FALSE;

1100     switch (Visitation)
1101     {
1102     case ASL_WALK_VISIT_DOWNWARD:

1104         while (Op)
1105         {
1106             if (!NodePreviouslyVisited)
1107             {
1108                 /* Let the callback process the node. */

1110                 Status = DescendingCallback (Op, Level, Context);
1111                 if (ACPI_SUCCESS (Status))
1112                 {
1113                     /* Visit children first, once */

1115                     if (Op->Asl.Child)
1116                     {
```



```
1117                         Level++;
1118                         Op = Op->Asl.Child;
1119                         continue;
1120                     }
1121                 }
1122                 else if (Status != AE_CTRL_DEPTH)
1123                 {
1124                     /* Exit immediately on any error */

1126                     return (Status);
1127                 }
1128             }

1130             /* Terminate walk at start op */

1132             if (Op == StartOp)
1133             {
1134                 break;
1135             }

1137             /* No more children, visit peers */

1139             if (Op->Asl.Next)
1140             {
1141                 Op = Op->Asl.Next;
1142                 NodePreviouslyVisited = FALSE;
1143             }
1144             else
1145             {
1146                 /* No children or peers, re-visit parent */

1148                 if (Level != 0 )
1149                 {
1150                     Level--;
1151                 }
1152                 Op = Op->Asl.Parent;
1153                 NodePreviouslyVisited = TRUE;
1154             }
1155         }
1156         break;

1158     case ASL_WALK_VISIT_UPWARD:

1160         while (Op)
1161         {
1162             /* Visit leaf node (no children) or parent node on return trip */

1164             if ((!Op->Asl.Child) ||
1165                 (NodePreviouslyVisited))
1166             {
1167                 /* Let the callback process the node. */

1169                 Status = AscendingCallback (Op, Level, Context);
1170                 if (ACPI_FAILURE (Status))
1171                 {
1172                     return (Status);
1173                 }
1174             }
1175             else
1176             {
1177                 /* Visit children first, once */

1179                 Level++;
1180                 Op = Op->Asl.Child;
1181                 continue;
1182             }
```

```
1184             /* Terminate walk at start op */

1186             if (Op == StartOp)
1187             {
1188                 break;
1189             }

1191             /* No more children, visit peers */

1193             if (Op->Asl.Next)
1194             {
1195                 Op = Op->Asl.Next;
1196                 NodePreviouslyVisited = FALSE;
1197             }
1198             else
1199             {
1200                 /* No children or peers, re-visit parent */

1202                 if (Level != 0 )
1203                 {
1204                     Level--;
1205                 }
1206                 Op = Op->Asl.Parent;
1207                 NodePreviouslyVisited = TRUE;
1208             }
1209         }
1210         break;

1212     case ASL_WALK_VISIT_TWICE:

1214         while (Op)
1215         {
1216             if (NodePreviouslyVisited)
1217             {
1218                 Status = AscendingCallback (Op, Level, Context);
1219                 if (ACPI_FAILURE (Status))
1220                 {
1221                     return (Status);
1222                 }
1223             }
1224             else
1225             {
1226                 /* Let the callback process the node. */

1228                 Status = DescendingCallback (Op, Level, Context);
1229                 if (ACPI_SUCCESS (Status))
1230                 {
1231                     /* Visit children first, once */

1233                     if (Op->Asl.Child)
1234                     {
1235                         Level++;
1236                         Op = Op->Asl.Child;
1237                         continue;
1238                     }
1239                 }
1240                 else if (Status != AE_CTRL_DEPTH)
1241                 {
1242                     /* Exit immediately on any error */

1244                     return (Status);
1245                 }
1246             }

1248             /* Terminate walk at start op */
```



```
1250             if (Op == StartOp)
1251             {
1252                 break;
1253             }

1255             /* No more children, visit peers */

1257             if (Op->Asl.Next)
1258             {
1259                 Op = Op->Asl.Next;
1260                 NodePreviouslyVisited = FALSE;
1261             }
1262             else
1263             {
1264                 /* No children or peers, re-visit parent */

1266                 if (Level != 0 )
1267                 {
1268                     Level--;
1269                 }
1270                 Op = Op->Asl.Parent;
1271                 NodePreviouslyVisited = TRUE;
1272             }
1273         }
1274         break;

1276     default:
1277         /* No other types supported */
1278         break;
1279     }

1281     /* If we get here, the walk completed with no errors */

1283     return (AE_OK);
1284 }
```

```
**********************************************************
    7356 Thu Dec 26 13:48:35 2013
new/usr/src/common/acpica/compiler/asltypes.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asltypes.h - compiler data types and struct definitions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ASLTYPES_H
#define __ASLTYPES_H


/******************************************************************************
 *
 * Structure definitions
 *
 *****************************************************************************/


/* Op flags for the ACPI_PARSE_OBJECT */

#define NODE_VISITED                0x00000001
#define NODE_AML_PACKAGE            0x00000002
#define NODE_IS_TARGET              0x00000004
```



```
#define NODE_IS_RESOURCE_DESC       0x00000008
#define NODE_IS_RESOURCE_FIELD      0x00000010
#define NODE_HAS_NO_EXIT            0x00000020
#define NODE_IF_HAS_NO_EXIT         0x00000040
#define NODE_NAME_INTERNALIZED      0x00000080
#define NODE_METHOD_NO_RETVAL       0x00000100
#define NODE_METHOD_SOME_NO_RETVAL  0x00000200
#define NODE_RESULT_NOT_USED        0x00000400
#define NODE_METHOD_TYPED           0x00000800
#define NODE_UNUSED_FLAG            0x00001000
#define NODE_COMPILE_TIME_CONST     0x00002000
#define NODE_IS_TERM_ARG            0x00004000
#define NODE_WAS_ONES_OP            0x00008000
#define NODE_IS_NAME_DECLARATION    0x00010000
#define NODE_COMPILER_EMITTED       0x00020000
#define NODE_IS_DUPLICATE           0x00040000
#define NODE_IS_RESOURCE_DATA       0x00080000
#define NODE_IS_NULL_RETURN         0x00100000

/* Keeps information about individual control methods */

typedef struct asl_method_info
{
    ACPI_PARSE_OBJECT       *Op;
    struct asl_method_info  *Next;
    UINT32                  ValidArgTypes[ACPI_METHOD_NUM_ARGS];
    UINT32                  ValidReturnTypes;
    UINT32                  NumReturnNoValue;
    UINT32                  NumReturnWithValue;
    UINT8                   NumArguments;
    UINT8                   LocalInitialized[ACPI_METHOD_NUM_LOCALS];
    UINT8                   ArgInitialized[ACPI_METHOD_NUM_ARGS];
    UINT8                   HasBeenTyped;
    UINT8                   ShouldBeSerialized;

} ASL_METHOD_INFO;


/* Parse tree walk info for control method analysis */

typedef struct asl_analysis_walk_info
{
    ASL_METHOD_INFO         *MethodStack;

} ASL_ANALYSIS_WALK_INFO;


/* An entry in the ParseOpcode to AmlOpcode mapping table */

typedef struct asl_mapping_entry
{
    UINT32                      Value;
    UINT32                      AcpiBtype;   /* Object type or return type */
    UINT16                      AmlOpcode;
    UINT8                       Flags;

} ASL_MAPPING_ENTRY;


/* Parse tree walk info structure */

typedef struct asl_walk_info
{
    ACPI_PARSE_OBJECT           **NodePtr;
    UINT32                      *LevelPtr;
```

```
} ASL_WALK_INFO;


/* File info */

typedef struct asl_file_info
{
    FILE                        *Handle;
    char                        *Filename;
    const char                  *ShortDescription;
    const char                  *Description;

} ASL_FILE_INFO;

typedef struct asl_file_status
{
    UINT32                  Line;
    UINT32                  Offset;

} ASL_FILE_STATUS;


/*
 * File types. Note: Any changes to this table must also be reflected
 * in the Gbl_Files array.
 */
typedef enum
{
    ASL_FILE_STDOUT             = 0,
    ASL_FILE_STDERR,
    ASL_FILE_INPUT,
    ASL_FILE_AML_OUTPUT,        /* Don't move these first 4 file types */
    ASL_FILE_SOURCE_OUTPUT,
    ASL_FILE_PREPROCESSOR,
    ASL_FILE_LISTING_OUTPUT,
    ASL_FILE_HEX_OUTPUT,
    ASL_FILE_NAMESPACE_OUTPUT,
    ASL_FILE_DEBUG_OUTPUT,
    ASL_FILE_ASM_SOURCE_OUTPUT,
    ASL_FILE_C_SOURCE_OUTPUT,
    ASL_FILE_ASM_INCLUDE_OUTPUT,
    ASL_FILE_C_INCLUDE_OUTPUT,
    ASL_FILE_C_OFFSET_OUTPUT

} ASL_FILE_TYPES;


#define ASL_MAX_FILE_TYPE       14
#define ASL_NUM_FILES           (ASL_MAX_FILE_TYPE + 1)


typedef struct asl_include_dir
{
    char                        *Dir;
    struct asl_include_dir      *Next;

} ASL_INCLUDE_DIR;


/* An entry in the exception list, one for each error/warning */

typedef struct asl_error_msg
{
    UINT32                      LineNumber;
    UINT32                      LogicalLineNumber;
    UINT32                      LogicalByteOffset;
```



```
    UINT32                      Column;
    char                        *Message;
    struct asl_error_msg        *Next;
    char                        *Filename;
    char                        *SourceLine;
    UINT32                      FilenameLength;
    UINT8                       MessageId;
    UINT8                       Level;

} ASL_ERROR_MSG;


/* An entry in the listing file stack (for include files) */

typedef struct asl_listing_node
{
    char                        *Filename;
    UINT32                      LineNumber;
    struct asl_listing_node     *Next;

} ASL_LISTING_NODE;


/* Callback interface for a parse tree walk */

/*
 * TBD - another copy of this is in adisasm.h, fix
 */
#ifndef ASL_WALK_CALLBACK_DEFINED
typedef
ACPI_STATUS (*ASL_WALK_CALLBACK) (
    ACPI_PARSE_OBJECT           *Op,
    UINT32                      Level,
    void                        *Context);
#define ASL_WALK_CALLBACK_DEFINED
#endif


typedef struct asl_event_info
{
    UINT64                      StartTime;
    UINT64                      EndTime;
    char                        *EventName;
    BOOLEAN                     Valid;

} ASL_EVENT_INFO;


#endif  /* __ASLTYPES_H */
```

```
**********************************************************
   28300 Thu Dec 26 13:48:35 2013
new/usr/src/common/acpica/compiler/aslutils.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslutils -- compiler utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acdisasm.h"
#include "acnamesp.h"
#include "amlcode.h"
#include <acapps.h>

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslutils")


/* Local prototypes */

static void
UtPadNameWithUnderscores (
    char                    *NameSeg,
```



```
    char                    *PaddedNameSeg);

static void
UtAttachNameseg (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name);


/******************************************************************************
 *
 * FUNCTION:    UtDisplaySupportedTables
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print all supported ACPI table names.
 *
 *****************************************************************************/

#define ACPI_TABLE_HELP_FORMAT  "%8u) %s    %s\n"

void
UtDisplaySupportedTables (
    void)
{
    ACPI_DMTABLE_DATA       *TableData;
    UINT32                  i;


    printf ("\nACPI tables supported by iASL version %8.8X:\n"
        "  (Compiler, Disassembler, Template Generator)\n\n",
        ACPI_CA_VERSION);

    /* Special tables */

    printf ("  Special tables and AML tables:\n");
    printf (ACPI_TABLE_HELP_FORMAT, 1, ACPI_RSDP_NAME, "Root System Description
    printf (ACPI_TABLE_HELP_FORMAT, 2, ACPI_SIG_FACS, "Firmware ACPI Control Str
    printf (ACPI_TABLE_HELP_FORMAT, 3, ACPI_SIG_DSDT, "Differentiated System Des
    printf (ACPI_TABLE_HELP_FORMAT, 4, ACPI_SIG_SSDT, "Secondary System Descript

    /* All data tables with common table header */

    printf ("\n  Standard ACPI data tables:\n");
    for (TableData = AcpiDmTableData, i = 5; TableData->Signature; TableData++,
    {
        printf (ACPI_TABLE_HELP_FORMAT, i, TableData->Signature, TableData->Name
    }
}


/******************************************************************************
 *
 * FUNCTION:    UtDisplayConstantOpcodes
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print AML opcodes that can be used in constant expressions.
 *
 *****************************************************************************/

void
UtDisplayConstantOpcodes (
```

```
    void)
{
    UINT32                  i;


    printf ("Constant expression opcode information\n\n");

    for (i = 0; i < sizeof (AcpiGbl_AmlOpInfo) / sizeof (ACPI_OPCODE_INFO); i++)
    {
        if (AcpiGbl_AmlOpInfo[i].Flags & AML_CONSTANT)
        {
            printf ("%s\n", AcpiGbl_AmlOpInfo[i].Name);
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    UtLocalCalloc
 *
 * PARAMETERS:  Size                - Bytes to be allocated
 *
 * RETURN:      Pointer to the allocated memory. Guaranteed to be valid.
 *
 * DESCRIPTION: Allocate zero-initialized memory. Aborts the compile on an
 *              allocation failure, on the assumption that nothing more can be
 *              accomplished.
 *
 *****************************************************************************/

void *
UtLocalCalloc (
    UINT32                  Size)
{
    void                    *Allocated;


    Allocated = ACPI_ALLOCATE_ZEROED (Size);
    if (!Allocated)
    {
        AslCommonError (ASL_ERROR, ASL_MSG_MEMORY_ALLOCATION,
            Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
            Gbl_InputByteCount, Gbl_CurrentColumn,
            Gbl_Files[ASL_FILE_INPUT].Filename, NULL);

        CmCleanupAndExit ();
        exit (1);
    }

    TotalAllocations++;
    TotalAllocated += Size;
    return (Allocated);
}


/******************************************************************************
 *
 * FUNCTION:    UtBeginEvent
 *
 * PARAMETERS:  Name                - Ascii name of this event
 *
 * RETURN:      Event number (integer index)
 *
 * DESCRIPTION: Saves the current time with this event
 *
```



```
 *****************************************************************************/

UINT8
UtBeginEvent (
    char                    *Name)
{

    if (AslGbl_NextEvent >= ASL_NUM_EVENTS)
    {
        AcpiOsPrintf ("Ran out of compiler event structs!\n");
        return (AslGbl_NextEvent);
    }

    /* Init event with current (start) time */

    AslGbl_Events[AslGbl_NextEvent].StartTime = AcpiOsGetTimer ();
    AslGbl_Events[AslGbl_NextEvent].EventName = Name;
    AslGbl_Events[AslGbl_NextEvent].Valid = TRUE;

    return (AslGbl_NextEvent++);
}


/******************************************************************************
 *
 * FUNCTION:    UtEndEvent
 *
 * PARAMETERS:  Event               - Event number (integer index)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Saves the current time (end time) with this event
 *
 *****************************************************************************/

void
UtEndEvent (
    UINT8                   Event)
{

    if (Event >= ASL_NUM_EVENTS)
    {
        return;
    }

    /* Insert end time for event */

    AslGbl_Events[Event].EndTime = AcpiOsGetTimer ();
}


/******************************************************************************
 *
 * FUNCTION:    UtHexCharToValue
 *
 * PARAMETERS:  HexChar             - Hex character in Ascii
 *
 * RETURN:      The binary value of the hex character
 *
 * DESCRIPTION: Perform ascii-to-hex translation
 *
 *****************************************************************************/

UINT8
UtHexCharToValue (
    int                     HexChar)
```

```
259 {
261     if (HexChar <= 0x39)
262     {
263         return ((UINT8) (HexChar - 0x30));
264     }
266     if (HexChar <= 0x46)
267     {
268         return ((UINT8) (HexChar - 0x37));
269     }
271     return ((UINT8) (HexChar - 0x57));
272 }


275 /******************************************************************************
276  *
277  * FUNCTION:    UtConvertByteToHex
278  *
279  * PARAMETERS:  RawByte             - Binary data
280  *              Buffer              - Pointer to where the hex bytes will be
281  *                                    stored
282  *
283  * RETURN:      Ascii hex byte is stored in Buffer.
284  *
285  * DESCRIPTION: Perform hex-to-ascii translation. The return data is prefixed
286  *              with "0x"
287  *
288  *****************************************************************************/

290 void
291 UtConvertByteToHex (
292     UINT8                   RawByte,
293     UINT8                   *Buffer)
294 {
296     Buffer[0] = '0';
297     Buffer[1] = 'x';
299     Buffer[2] = (UINT8) AslHexLookup[(RawByte >> 4) & 0xF];
300     Buffer[3] = (UINT8) AslHexLookup[RawByte & 0xF];
301 }


304 /******************************************************************************
305  *
306  * FUNCTION:    UtConvertByteToAsmHex
307  *
308  * PARAMETERS:  RawByte             - Binary data
309  *              Buffer              - Pointer to where the hex bytes will be
310  *                                    stored
311  *
312  * RETURN:      Ascii hex byte is stored in Buffer.
313  *
314  * DESCRIPTION: Perform hex-to-ascii translation. The return data is prefixed
315  *              with "0x"
316  *
317  *****************************************************************************/

319 void
320 UtConvertByteToAsmHex (
321     UINT8                   RawByte,
322     UINT8                   *Buffer)
323 {
```



```
325     Buffer[0] = '0';
326     Buffer[1] = (UINT8) AslHexLookup[(RawByte >> 4) & 0xF];
327     Buffer[2] = (UINT8) AslHexLookup[RawByte & 0xF];
328     Buffer[3] = 'h';
329 }


332 /******************************************************************************
333  *
334  * FUNCTION:    DbgPrint
335  *
336  * PARAMETERS:  Type                - Type of output
337  *              Fmt                 - Printf format string
338  *              ...                 - variable printf list
339  *
340  * RETURN:      None
341  *
342  * DESCRIPTION: Conditional print statement. Prints to stderr only if the
343  *              debug flag is set.
344  *
345  *****************************************************************************/

347 void
348 DbgPrint (
349     UINT32                  Type,
350     char                    *Fmt,
351     ...)
352 {
353     va_list                 Args;


356     if (!Gbl_DebugFlag)
357     {
358         return;
359     }
361     if ((Type == ASL_PARSE_OUTPUT) &&
362         (!(AslCompilerdebug)))
363     {
364         return;
365     }
367     va_start (Args, Fmt);
368     (void) vfprintf (stderr, Fmt, Args);
369     va_end (Args);
370     return;
371 }


374 /******************************************************************************
375  *
376  * FUNCTION:    UtPrintFormattedName
377  *
378  * PARAMETERS:  ParseOpcode         - Parser keyword ID
379  *              Level               - Indentation level
380  *
381  * RETURN:      None
382  *
383  * DESCRIPTION: Print the ascii name of the parse opcode.
384  *
385  *****************************************************************************/

387 #define TEXT_OFFSET 10

389 void
390 UtPrintFormattedName (
```

```
391     UINT16                  ParseOpcode,
392     UINT32                  Level)
393 {

395     if (Level)
396     {
397         DbgPrint (ASL_TREE_OUTPUT,
398             "%*s", (3 * Level), " ");
399     }
400     DbgPrint (ASL_TREE_OUTPUT,
401         " %-20.20s", UtGetOpName (ParseOpcode));

403     if (Level < TEXT_OFFSET)
404     {
405         DbgPrint (ASL_TREE_OUTPUT,
406             "%*s", (TEXT_OFFSET - Level) * 3, " ");
407     }
408 }


411 /*******************************************************************************
412  *
413  * FUNCTION:    UtSetParseOpName
414  *
415  * PARAMETERS:  Op                  - Parse op to be named.
416  *
417  * RETURN:      None
418  *
419  * DESCRIPTION: Insert the ascii name of the parse opcode
420  *
421  ******************************************************************************/

423 void
424 UtSetParseOpName (
425     ACPI_PARSE_OBJECT       *Op)
426 {

428     strncpy (Op->Asl.ParseOpName, UtGetOpName (Op->Asl.ParseOpcode),
429         ACPI_MAX_PARSEOP_NAME);
430 }


433 /*******************************************************************************
434  *
435  * FUNCTION:    UtDisplaySummary
436  *
437  * PARAMETERS:  FileID              - ID of outpout file
438  *
439  * RETURN:      None
440  *
441  * DESCRIPTION: Display compilation statistics
442  *
443  ******************************************************************************/

445 void
446 UtDisplaySummary (
447     UINT32                  FileId)
448 {
449     UINT32                  i;


452     if (FileId != ASL_FILE_STDOUT)
453     {
454         /* Compiler name and version number */

456         FlPrintFile (FileId, "%s version %X%s [%s]\n\n",
```



```
457             ASL_COMPILER_NAME, (UINT32) ACPI_CA_VERSION, ACPI_WIDTH, __DATE__);
458     }

460     /* Summary of main input and output files */

462     if (Gbl_FileType == ASL_INPUT_TYPE_ASCII_DATA)
463     {
464         FlPrintFile (FileId,
465             "%-14s %s - %u lines, %u bytes, %u fields\n",
466             "Table Input:",
467             Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_CurrentLineNumber,
468             Gbl_InputByteCount, Gbl_InputFieldCount);

470         if ((Gbl_ExceptionCount[ASL_ERROR] == 0) || (Gbl_IgnoreErrors))
471         {
472             FlPrintFile (FileId,
473                 "%-14s %s - %u bytes\n",
474                 "Binary Output:",
475                 Gbl_Files[ASL_FILE_AML_OUTPUT].Filename, Gbl_TableLength);
476         }
477     }
478     else
479     {
480         FlPrintFile (FileId,
481             "%-14s %s - %u lines, %u bytes, %u keywords\n",
482             "ASL Input:",
483             Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_CurrentLineNumber,
484             Gbl_InputByteCount, TotalKeywords);

486         /* AML summary */

488         if ((Gbl_ExceptionCount[ASL_ERROR] == 0) || (Gbl_IgnoreErrors))
489         {
490             FlPrintFile (FileId,
491                 "%-14s %s - %u bytes, %u named objects, %u executable opcodes\n"
492                 "AML Output:",
493                 Gbl_Files[ASL_FILE_AML_OUTPUT].Filename, Gbl_TableLength,
494                 TotalNamedObjects, TotalExecutableOpcodes);
495         }
496     }

498     /* Display summary of any optional files */

500     for (i = ASL_FILE_SOURCE_OUTPUT; i <= ASL_MAX_FILE_TYPE; i++)
501     {
502         if (!Gbl_Files[i].Filename || !Gbl_Files[i].Handle)
503         {
504             continue;
505         }

507         /* .SRC is a temp file unless specifically requested */

509         if ((i == ASL_FILE_SOURCE_OUTPUT) && (!Gbl_SourceOutputFlag))
510         {
511             continue;
512         }

514         /* .I is a temp file unless specifically requested */

516         if ((i == ASL_FILE_PREPROCESSOR) && (!Gbl_PreprocessorOutputFlag))
517         {
518             continue;
519         }

521         FlPrintFile (FileId, "%14s %s - %u bytes\n",
522             Gbl_Files[i].ShortDescription,
```

```
523             Gbl_Files[i].Filename, FlGetFileSize (i));
524     }

526     /* Error summary */

528     FlPrintFile (FileId,
529         "\nCompilation complete. %u Errors, %u Warnings, %u Remarks",
530         Gbl_ExceptionCount[ASL_ERROR],
531         Gbl_ExceptionCount[ASL_WARNING] +
532             Gbl_ExceptionCount[ASL_WARNING2] +
533             Gbl_ExceptionCount[ASL_WARNING3],
534         Gbl_ExceptionCount[ASL_REMARK]);

536     if (Gbl_FileType != ASL_INPUT_TYPE_ASCII_DATA)
537     {
538         FlPrintFile (FileId,
539             ", %u Optimizations", Gbl_ExceptionCount[ASL_OPTIMIZATION]);
540     }

542     FlPrintFile (FileId, "\n");
543 }


546 /*******************************************************************************
547  *
548  * FUNCTION:    UtCheckIntegerRange
549  *
550  * PARAMETERS:  Op                  - Integer parse node
551  *              LowValue            - Smallest allowed value
552  *              HighValue           - Largest allowed value
553  *
554  * RETURN:      Op if OK, otherwise NULL
555  *
556  * DESCRIPTION: Check integer for an allowable range
557  *
558  ******************************************************************************/

560 ACPI_PARSE_OBJECT *
561 UtCheckIntegerRange (
562     ACPI_PARSE_OBJECT       *Op,
563     UINT32                  LowValue,
564     UINT32                  HighValue)
565 {

567     if (!Op)
568     {
569         return (NULL);
570     }

572     if ((Op->Asl.Value.Integer < LowValue) ||
573         (Op->Asl.Value.Integer > HighValue))
574     {
575         sprintf (MsgBuffer, "0x%X, allowable: 0x%X-0x%X",
576             (UINT32) Op->Asl.Value.Integer, LowValue, HighValue);

578         AslError (ASL_ERROR, ASL_MSG_RANGE, Op, MsgBuffer);
579         return (NULL);
580     }

582     return (Op);
583 }


586 /*******************************************************************************
587  *
588  * FUNCTION:    UtGetStringBuffer
```



```
589  *
590  * PARAMETERS:  Length              - Size of buffer requested
591  *
592  * RETURN:      Pointer to the buffer. Aborts on allocation failure
593  *
594  * DESCRIPTION: Allocate a string buffer. Bypass the local
595  *              dynamic memory manager for performance reasons (This has a
596  *              major impact on the speed of the compiler.)
597  *
598  ******************************************************************************/

600 char *
601 UtGetStringBuffer (
602     UINT32                  Length)
603 {
604     char                    *Buffer;


607     if ((Gbl_StringCacheNext + Length) >= Gbl_StringCacheLast)
608     {
609         Gbl_StringCacheNext = UtLocalCalloc (ASL_STRING_CACHE_SIZE + Length);
610         Gbl_StringCacheLast = Gbl_StringCacheNext + ASL_STRING_CACHE_SIZE +
611                                 Length;
612     }

614     Buffer = Gbl_StringCacheNext;
615     Gbl_StringCacheNext += Length;

617     return (Buffer);
618 }


621 /*******************************************************************************
622  *
623  * FUNCTION:    UtExpandLineBuffers
624  *
625  * PARAMETERS:  None. Updates global line buffer pointers.
626  *
627  * RETURN:      None. Reallocates the global line buffers
628  *
629  * DESCRIPTION: Called if the current line buffer becomes filled. Reallocates
630  *              all global line buffers and updates Gbl_LineBufferSize. NOTE:
631  *              Also used for the initial allocation of the buffers, when
632  *              all of the buffer pointers are NULL. Initial allocations are
633  *              of size ASL_DEFAULT_LINE_BUFFER_SIZE
634  *
635  ******************************************************************************/

637 void
638 UtExpandLineBuffers (
639     void)
640 {
641     UINT32                  NewSize;


644     /* Attempt to double the size of all line buffers */

646     NewSize = Gbl_LineBufferSize * 2;
647     if (Gbl_CurrentLineBuffer)
648     {
649         DbgPrint (ASL_DEBUG_OUTPUT,"Increasing line buffer size from %u to %u\n"
650             Gbl_LineBufferSize, NewSize);
651     }

653     Gbl_CurrentLineBuffer = realloc (Gbl_CurrentLineBuffer, NewSize);
654     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;
```

```
655     if (!Gbl_CurrentLineBuffer)
656     {
657         goto ErrorExit;
658     }

660     Gbl_MainTokenBuffer = realloc (Gbl_MainTokenBuffer, NewSize);
661     if (!Gbl_MainTokenBuffer)
662     {
663         goto ErrorExit;
664     }

666     Gbl_MacroTokenBuffer = realloc (Gbl_MacroTokenBuffer, NewSize);
667     if (!Gbl_MacroTokenBuffer)
668     {
669         goto ErrorExit;
670     }

672     Gbl_ExpressionTokenBuffer = realloc (Gbl_ExpressionTokenBuffer, NewSize);
673     if (!Gbl_ExpressionTokenBuffer)
674     {
675         goto ErrorExit;
676     }

678     Gbl_LineBufferSize = NewSize;
679     return;


682     /* On error above, simply issue error messages and abort, cannot continue */

684 ErrorExit:
685     printf ("Could not increase line buffer size from %u to %u\n",
686         Gbl_LineBufferSize, Gbl_LineBufferSize * 2);

688     AslError (ASL_ERROR, ASL_MSG_BUFFER_ALLOCATION,
689         NULL, NULL);
690     AslAbort ();
691 }


694 /*******************************************************************************
695  *
696  * FUNCTION:    UtInternalizeName
697  *
698  * PARAMETERS:  ExternalName        - Name to convert
699  *              ConvertedName       - Where the converted name is returned
700  *
701  * RETURN:      Status
702  *
703  * DESCRIPTION: Convert an external (ASL) name to an internal (AML) name
704  *
705  ******************************************************************************/

707 ACPI_STATUS
708 UtInternalizeName (
709     char                    *ExternalName,
710     char                    **ConvertedName)
711 {
712     ACPI_NAMESTRING_INFO    Info;
713     ACPI_STATUS             Status;


716     if (!ExternalName)
717     {
718         return (AE_OK);
719     }
```



```
721     /* Get the length of the new internal name */

723     Info.ExternalName = ExternalName;
724     AcpiNsGetInternalNameLength (&Info);

726     /* We need a segment to store the internal  name */

728     Info.InternalName = UtGetStringBuffer (Info.Length);
729     if (!Info.InternalName)
730     {
731         return (AE_NO_MEMORY);
732     }

734     /* Build the name */

736     Status = AcpiNsBuildInternalName (&Info);
737     if (ACPI_FAILURE (Status))
738     {
739         return (Status);
740     }

742     *ConvertedName = Info.InternalName;
743     return (AE_OK);
744 }


747 /*******************************************************************************
748  *
749  * FUNCTION:    UtPadNameWithUnderscores
750  *
751  * PARAMETERS:  NameSeg             - Input nameseg
752  *              PaddedNameSeg       - Output padded nameseg
753  *
754  * RETURN:      Padded nameseg.
755  *
756  * DESCRIPTION: Pads a NameSeg with underscores if necessary to form a full
757  *              ACPI_NAME.
758  *
759  ******************************************************************************/

761 static void
762 UtPadNameWithUnderscores (
763     char                    *NameSeg,
764     char                    *PaddedNameSeg)
765 {
766     UINT32                  i;


769     for (i = 0; (i < ACPI_NAME_SIZE); i++)
770     {
771         if (*NameSeg)
772         {
773             *PaddedNameSeg = *NameSeg;
774             NameSeg++;
775         }
776         else
777         {
778             *PaddedNameSeg = '_';
779         }
780         PaddedNameSeg++;
781     }
782 }


785 /*******************************************************************************
786  *
```

```
 * FUNCTION:    UtAttachNameseg
 *
 * PARAMETERS:  Op                  - Parent parse node
 *              Name                - Full ExternalName
 *
 * RETURN:      None; Sets the NameSeg field in parent node
 *
 * DESCRIPTION: Extract the last nameseg of the ExternalName and store it
 *              in the NameSeg field of the Op.
 *
 *****************************************************************************/

static void
UtAttachNameseg (
    ACPI_PARSE_OBJECT       *Op,
    char                    *Name)
{
    char                    *NameSeg;
    char                    PaddedNameSeg[4];


    if (!Name)
    {
        return;
    }

    /* Look for the last dot in the namepath */

    NameSeg = strrchr (Name, '.');
    if (NameSeg)
    {
        /* Found last dot, we have also found the final nameseg */

        NameSeg++;
        UtPadNameWithUnderscores (NameSeg, PaddedNameSeg);
    }
    else
    {
        /* No dots in the namepath, there is only a single nameseg. */
        /* Handle prefixes */

        while (ACPI_IS_ROOT_PREFIX (*Name) ||
               ACPI_IS_PARENT_PREFIX (*Name))
        {
            Name++;
        }

        /* Remaining string should be one single nameseg */

        UtPadNameWithUnderscores (Name, PaddedNameSeg);
    }

    ACPI_MOVE_NAME (Op->Asl.NameSeg, PaddedNameSeg);
}


/*****************************************************************************
 *
 * FUNCTION:    UtAttachNamepathToOwner
 *
 * PARAMETERS:  Op                  - Parent parse node
 *              NameOp              - Node that contains the name
 *
 * RETURN:      Sets the ExternalName and Namepath in the parent node
 *
 * DESCRIPTION: Store the name in two forms in the parent node: The original
```



```
 *              (external) name, and the internalized name that is used within
 *              the ACPI namespace manager.
 *
 *****************************************************************************/

void
UtAttachNamepathToOwner (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_PARSE_OBJECT       *NameOp)
{
    ACPI_STATUS             Status;


    /* Full external path */

    Op->Asl.ExternalName = NameOp->Asl.Value.String;

    /* Save the NameOp for possible error reporting later */

    Op->Asl.ParentMethod = (void *) NameOp;

    /* Last nameseg of the path */

    UtAttachNameseg (Op, Op->Asl.ExternalName);

    /* Create internalized path */

    Status = UtInternalizeName (NameOp->Asl.Value.String, &Op->Asl.Namepath);
    if (ACPI_FAILURE (Status))
    {
        /* TBD: abort on no memory */
    }
}


/*******************************************************************************
 *
 * FUNCTION:    UtDoConstant
 *
 * PARAMETERS:  String              - Hex, Octal, or Decimal string
 *
 * RETURN:      Converted Integer
 *
 * DESCRIPTION: Convert a string to an integer, with error checking.
 *
 ******************************************************************************/

UINT64
UtDoConstant (
    char                    *String)
{
    ACPI_STATUS             Status;
    UINT64                  Converted;
    char                    ErrBuf[64];


    Status = UtStrtoul64 (String, 0, &Converted);
    if (ACPI_FAILURE (Status))
    {
        sprintf (ErrBuf, "%s %s\n", "Conversion error:",
            AcpiFormatException (Status));
        AslCompilererror (ErrBuf);
    }

    return (Converted);
}
```

```
/* TBD: use version in ACPI CA main code base? */

/*******************************************************************************
 *
 * FUNCTION:    UtStrtoul64
 *
 * PARAMETERS:  String              - Null terminated string
 *              Terminater          - Where a pointer to the terminating byte
 *                                    is returned
 *              Base                - Radix of the string
 *
 * RETURN:      Converted value
 *
 * DESCRIPTION: Convert a string into an unsigned value.
 *
 ******************************************************************************/

ACPI_STATUS
UtStrtoul64 (
    char                    *String,
    UINT32                  Base,
    UINT64                  *RetInteger)
{
    UINT32                  Index;
    UINT32                  Sign;
    UINT64                  ReturnValue = 0;
    ACPI_STATUS             Status = AE_OK;


    *RetInteger = 0;

    switch (Base)
    {
    case 0:
    case 8:
    case 10:
    case 16:

        break;

    default:
        /*
         * The specified Base parameter is not in the domain of
         * this function:
         */
        return (AE_BAD_PARAMETER);
    }

    /* Skip over any white space in the buffer: */

    while (isspace ((int) *String) || *String == '\t')
    {
        ++String;
    }

    /*
     * The buffer may contain an optional plus or minus sign.
     * If it does, then skip over it but remember what is was:
     */
    if (*String == '-')
    {
        Sign = NEGATIVE;
        ++String;
    }
```



```
    else if (*String == '+')
    {
        ++String;
        Sign = POSITIVE;
    }
    else
    {
        Sign = POSITIVE;
    }

    /*
     * If the input parameter Base is zero, then we need to
     * determine if it is octal, decimal, or hexadecimal:
     */
    if (Base == 0)
    {
        if (*String == '0')
        {
            if (tolower ((int) *(++String)) == 'x')
            {
                Base = 16;
                ++String;
            }
            else
            {
                Base = 8;
            }
        }
        else
        {
            Base = 10;
        }
    }

    /*
     * For octal and hexadecimal bases, skip over the leading
     * 0 or 0x, if they are present.
     */
    if (Base == 8 && *String == '0')
    {
        String++;
    }

    if (Base == 16 &&
        *String == '0' &&
        tolower ((int) *(++String)) == 'x')
    {
        String++;
    }

    /* Main loop: convert the string to an unsigned long */

    while (*String)
    {
        if (isdigit ((int) *String))
        {
            Index = ((UINT8) *String) - '0';
        }
        else
        {
            Index = (UINT8) toupper ((int) *String);
            if (isupper ((int) Index))
            {
                Index = Index - 'A' + 10;
            }
            else
```

```
1051             {
1052                 goto ErrorExit;
1053             }
1054         }

1056         if (Index >= Base)
1057         {
1058             goto ErrorExit;
1059         }

1061         /* Check to see if value is out of range: */

1063         if (ReturnValue > ((ACPI_UINT64_MAX - (UINT64) Index) /
1064                            (UINT64) Base))
1065         {
1066             goto ErrorExit;
1067         }
1068         else
1069         {
1070             ReturnValue *= Base;
1071             ReturnValue += Index;
1072         }

1074         ++String;
1075     }


1078     /* If a minus sign was present, then "the conversion is negated": */

1080     if (Sign == NEGATIVE)
1081     {
1082         ReturnValue = (ACPI_UINT32_MAX - ReturnValue) + 1;
1083     }

1085     *RetInteger = ReturnValue;
1086     return (Status);


1089 ErrorExit:
1090     switch (Base)
1091     {
1092     case 8:

1094         Status = AE_BAD_OCTAL_CONSTANT;
1095         break;

1097     case 10:

1099         Status = AE_BAD_DECIMAL_CONSTANT;
1100         break;

1102     case 16:

1104         Status = AE_BAD_HEX_CONSTANT;
1105         break;

1107     default:

1109         /* Base validated above */

1111         break;
1112     }

1114     return (Status);
1115 }
```

```
**********************************************************
    6676 Thu Dec 26 13:48:36 2013
new/usr/src/common/acpica/compiler/asluuid.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asluuid-- compiler UUID support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("asluuid")


/*
 * UUID support functions.
 *
 * This table is used to convert an input UUID ascii string to a 16 byte
 * buffer and the reverse. The table maps a UUID buffer index 0-15 to
 * the index within the 36-byte UUID string where the associated 2-byte
 * hex value can be found.
 *
 * 36-byte UUID strings are of the form:
 *     aabbccdd-eeff-gghh-iijj-kkllmmnnoopp
```



```
 * Where aa-pp are one byte hex numbers, made up of two hex digits
 *
 * Note: This table is basically the inverse of the string-to-offset table
 * found in the ACPI spec in the description of the ToUUID macro.
 */
static UINT8    Gbl_MapToUuidOffset[16] =
{
    6,4,2,0,11,9,16,14,19,21,24,26,28,30,32,34
};

#define UUID_BUFFER_LENGTH          16
#define UUID_STRING_LENGTH          36

/* Positions for required hyphens (dashes) in UUID strings */

#define UUID_HYPHEN1_OFFSET         8
#define UUID_HYPHEN2_OFFSET         13
#define UUID_HYPHEN3_OFFSET         18
#define UUID_HYPHEN4_OFFSET         23


/******************************************************************************
 *
 * FUNCTION:    AuValiduateUuid
 *
 * PARAMETERS:  InString            - 36-byte formatted UUID string
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check all 36 characters for correct format
 *
 *****************************************************************************/

ACPI_STATUS
AuValidateUuid (
    char                    *InString)
{
    UINT32                  i;


    if (!InString || (ACPI_STRLEN (InString) != UUID_STRING_LENGTH))
    {
        return (AE_BAD_PARAMETER);
    }

    /* Check all 36 characters for correct format */

    for (i = 0; i < UUID_STRING_LENGTH; i++)
    {
        /* Must have 4 hyphens (dashes) in these positions: */

        if ((i == UUID_HYPHEN1_OFFSET) ||
            (i == UUID_HYPHEN2_OFFSET) ||
            (i == UUID_HYPHEN3_OFFSET) ||
            (i == UUID_HYPHEN4_OFFSET))
        {
            if (InString[i] != '-')
            {
                return (AE_BAD_PARAMETER);
            }
        }

        /* All other positions must contain hex digits */

        else
        {
```

```
127             if (!isxdigit ((int) InString[i]))
128             {
129                 return (AE_BAD_PARAMETER);
130             }
131         }
132     }

134     return (AE_OK);
135 }


138 /*******************************************************************************
139  *
140  * FUNCTION:    AuConvertStringToUuid
141  *
142  * PARAMETERS:  InString            - 36-byte formatted UUID string
143  *              UuidBuffer          - 16-byte UUID buffer
144  *
145  * RETURN:      Status
146  *
147  * DESCRIPTION: Convert 36-byte formatted UUID string to 16-byte UUID buffer
148  *
149  ******************************************************************************/

151 ACPI_STATUS
152 AuConvertStringToUuid (
153     char                    *InString,
154     char                    *UuidBuffer)
155 {
156     UINT32                  i;


159     if (!InString || !UuidBuffer)
160     {
161         return (AE_BAD_PARAMETER);
162     }

164     for (i = 0; i < UUID_BUFFER_LENGTH; i++)
165     {
166         UuidBuffer[i]  = (char) (UtHexCharToValue (InString[Gbl_MapToUuidOffset[
167         UuidBuffer[i] |= (char)  UtHexCharToValue (InString[Gbl_MapToUuidOffset[
168     }

170     return (AE_OK);
171 }


174 /*******************************************************************************
175  *
176  * FUNCTION:    AuConvertUuidToString
177  *
178  * PARAMETERS:  UuidBuffer          - 16-byte UUID buffer
179  *              OutString           - 36-byte formatted UUID string
180  *
181  * RETURN:      Status
182  *
183  * DESCRIPTION: Convert 16-byte UUID buffer to 36-byte formatted UUID string
184  *              OutString must be 37 bytes to include null terminator.
185  *
186  ******************************************************************************/

188 ACPI_STATUS
189 AuConvertUuidToString (
190     char                    *UuidBuffer,
191     char                    *OutString)
192 {
```



```
193     UINT32                  i;


196     if (!UuidBuffer || !OutString)
197     {
198         return (AE_BAD_PARAMETER);
199     }

201     for (i = 0; i < UUID_BUFFER_LENGTH; i++)
202     {
203         OutString[Gbl_MapToUuidOffset[i]] =     (UINT8) AslHexLookup[(UuidBuffer
204         OutString[Gbl_MapToUuidOffset[i] + 1] = (UINT8) AslHexLookup[UuidBuffer[
205     }

207     /* Insert required hyphens (dashes) */

209     OutString[UUID_HYPHEN1_OFFSET] =
210     OutString[UUID_HYPHEN2_OFFSET] =
211     OutString[UUID_HYPHEN3_OFFSET] =
212     OutString[UUID_HYPHEN4_OFFSET] = '-';

214     OutString[UUID_STRING_LENGTH] = 0; /* Null terminate */
215     return (AE_OK);
216 }
```

```
**********************************************************
   20946 Thu Dec 26 13:48:36 2013
new/usr/src/common/acpica/compiler/aslwalks.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslwalks.c - Miscellaneous analytical parse tree walks
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acparser.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslwalks")


/*******************************************************************************
 *
 * FUNCTION:    AnMethodTypingWalkEnd
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
```



```
 * RETURN:      Status
 *
 * DESCRIPTION: Ascending callback for typing walk. Complete the method
 *              return analysis. Check methods for:
 *              1) Initialized local variables
 *              2) Valid arguments
 *              3) Return types
 *
 ******************************************************************************/

ACPI_STATUS
AnMethodTypingWalkEnd (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    UINT32                  ThisNodeBtype;


    switch (Op->Asl.ParseOpcode)
    {
    case PARSEOP_METHOD:

        Op->Asl.CompileFlags |= NODE_METHOD_TYPED;
        break;

    case PARSEOP_RETURN:

        if ((Op->Asl.Child) &&
            (Op->Asl.Child->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG))
        {
            ThisNodeBtype = AnGetBtype (Op->Asl.Child);

            if ((Op->Asl.Child->Asl.ParseOpcode == PARSEOP_METHODCALL) &&
                (ThisNodeBtype == (ACPI_UINT32_MAX -1)))
            {
                /*
                 * The called method is untyped at this time (typically a
                 * forward reference).
                 *
                 * Check for a recursive method call first.
                 */
                if (Op->Asl.ParentMethod != Op->Asl.Child->Asl.Node->Op)
                {
                    /* We must type the method here */

                    TrWalkParseTree (Op->Asl.Child->Asl.Node->Op,
                        ASL_WALK_VISIT_UPWARD, NULL,
                        AnMethodTypingWalkEnd, NULL);

                    ThisNodeBtype = AnGetBtype (Op->Asl.Child);
                }
            }

            /* Returns a value, save the value type */

            if (Op->Asl.ParentMethod)
            {
                Op->Asl.ParentMethod->Asl.AcpiBtype |= ThisNodeBtype;
            }
        }
        break;

    default:

        break;
```

```
127     }
129     return (AE_OK);
130 }


133 /*******************************************************************************
134  *
135  * FUNCTION:    AnOperandTypecheckWalkEnd
136  *
137  * PARAMETERS:  ASL_WALK_CALLBACK
138  *
139  * RETURN:      Status
140  *
141  * DESCRIPTION: Ascending callback for analysis walk. Complete method
142  *              return analysis.
143  *
144  ******************************************************************************/

146 ACPI_STATUS
147 AnOperandTypecheckWalkEnd (
148     ACPI_PARSE_OBJECT       *Op,
149     UINT32                  Level,
150     void                    *Context)
151 {
152     const ACPI_OPCODE_INFO  *OpInfo;
153     UINT32                  RuntimeArgTypes;
154     UINT32                  RuntimeArgTypes2;
155     UINT32                  RequiredBtypes;
156     UINT32                  ThisNodeBtype;
157     UINT32                  CommonBtypes;
158     UINT32                  OpcodeClass;
159     ACPI_PARSE_OBJECT       *ArgOp;
160     UINT32                  ArgType;


163     switch (Op->Asl.AmlOpcode)
164     {
165     case AML_RAW_DATA_BYTE:
166     case AML_RAW_DATA_WORD:
167     case AML_RAW_DATA_DWORD:
168     case AML_RAW_DATA_QWORD:
169     case AML_RAW_DATA_BUFFER:
170     case AML_RAW_DATA_CHAIN:
171     case AML_PACKAGE_LENGTH:
172     case AML_UNASSIGNED_OPCODE:
173     case AML_DEFAULT_ARG_OP:

175         /* Ignore the internal (compiler-only) AML opcodes */

177         return (AE_OK);

179     default:

181         break;
182     }

184     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
185     if (!OpInfo)
186     {
187         return (AE_OK);
188     }

190     ArgOp           = Op->Asl.Child;
191     RuntimeArgTypes = OpInfo->RuntimeArgs;
192     OpcodeClass     = OpInfo->Class;
```



```
194 #ifdef ASL_ERROR_NAMED_OBJECT_IN_WHILE
195     /*
196      * Update 11/2008: In practice, we can't perform this check. A simple
197      * analysis is not sufficient. Also, it can cause errors when compiling
198      * disassembled code because of the way Switch operators are implemented
199      * (a While(One) loop with a named temp variable created within.)
200      */

202     /*
203      * If we are creating a named object, check if we are within a while loop
204      * by checking if the parent is a WHILE op. This is a simple analysis, but
205      * probably sufficient for many cases.
206      *
207      * Allow Scope(), Buffer(), and Package().
208      */
209     if (((OpcodeClass == AML_CLASS_NAMED_OBJECT) && (Op->Asl.AmlOpcode != AML_SC
210          ((OpcodeClass == AML_CLASS_CREATE) && (OpInfo->Flags & AML_NSNODE)))
211     {
212         if (Op->Asl.Parent->Asl.AmlOpcode == AML_WHILE_OP)
213         {
214             AslError (ASL_ERROR, ASL_MSG_NAMED_OBJECT_IN_WHILE, Op, NULL);
215         }
216     }
217 #endif

219     /*
220      * Special case for control opcodes IF/RETURN/WHILE since they
221      * have no runtime arg list (at this time)
222      */
223     switch (Op->Asl.AmlOpcode)
224     {
225     case AML_IF_OP:
226     case AML_WHILE_OP:
227     case AML_RETURN_OP:

229         if (ArgOp->Asl.ParseOpcode == PARSEOP_METHODCALL)
230         {
231             /* Check for an internal method */

233             if (AnIsInternalMethod (ArgOp))
234             {
235                 return (AE_OK);
236             }

238             /* The lone arg is a method call, check it */

240             RequiredBtypes = AnMapArgTypeToBtype (ARGI_INTEGER);
241             if (Op->Asl.AmlOpcode == AML_RETURN_OP)
242             {
243                 RequiredBtypes = 0xFFFFFFFF;
244             }

246             ThisNodeBtype = AnGetBtype (ArgOp);
247             if (ThisNodeBtype == ACPI_UINT32_MAX)
248             {
249                 return (AE_OK);
250             }
251             AnCheckMethodReturnValue (Op, OpInfo, ArgOp,
252                 RequiredBtypes, ThisNodeBtype);
253         }
254         return (AE_OK);

256     default:

258         break;
```

```
259     }
261     /* Ignore the non-executable opcodes */
263     if (RuntimeArgTypes == ARGI_INVALID_OPCODE)
264     {
265         return (AE_OK);
266     }
268     switch (OpcodeClass)
269     {
270     case AML_CLASS_EXECUTE:
271     case AML_CLASS_CREATE:
272     case AML_CLASS_CONTROL:
273     case AML_CLASS_RETURN_VALUE:
275         /* TBD: Change class or fix typechecking for these */
277         if ((Op->Asl.AmlOpcode == AML_BUFFER_OP)        ||
278             (Op->Asl.AmlOpcode == AML_PACKAGE_OP)       ||
279             (Op->Asl.AmlOpcode == AML_VAR_PACKAGE_OP))
280         {
281             break;
282         }
284         /* Reverse the runtime argument list */
286         RuntimeArgTypes2 = 0;
287         while ((ArgType = GET_CURRENT_ARG_TYPE (RuntimeArgTypes)))
288         {
289             RuntimeArgTypes2 <<= ARG_TYPE_WIDTH;
290             RuntimeArgTypes2 |= ArgType;
291             INCREMENT_ARG_LIST (RuntimeArgTypes);
292         }
294         while ((ArgType = GET_CURRENT_ARG_TYPE (RuntimeArgTypes2)))
295         {
296             RequiredBtypes = AnMapArgTypeToBtype (ArgType);
298             ThisNodeBtype = AnGetBtype (ArgOp);
299             if (ThisNodeBtype == ACPI_UINT32_MAX)
300             {
301                 goto NextArgument;
302             }
304             /* Examine the arg based on the required type of the arg */
306             switch (ArgType)
307             {
308             case ARGI_TARGETREF:
310                 if (ArgOp->Asl.ParseOpcode == PARSEOP_ZERO)
311                 {
312                     /* ZERO is the placeholder for "don't store result" */
314                     ThisNodeBtype = RequiredBtypes;
315                     break;
316                 }
318                 if (ArgOp->Asl.ParseOpcode == PARSEOP_INTEGER)
319                 {
320                     /*
321                      * This is the case where an original reference to a resourc
322                      * descriptor field has been replaced by an (Integer) offset
323                      * These named fields are supported at compile-time only;
324                      * the names are not passed to the interpreter (via the AML)
```



```
325                      */
326                     if ((ArgOp->Asl.Node->Type == ACPI_TYPE_LOCAL_RESOURCE_FIELD
327                         (ArgOp->Asl.Node->Type == ACPI_TYPE_LOCAL_RESOURCE))
328                     {
329                         AslError (ASL_ERROR, ASL_MSG_RESOURCE_FIELD, ArgOp, NULL
330                     }
331                     else
332                     {
333                         AslError (ASL_ERROR, ASL_MSG_INVALID_TYPE, ArgOp, NULL);
334                     }
335                     break;
336                 }
338                 if ((ArgOp->Asl.ParseOpcode == PARSEOP_METHODCALL) ||
339                     (ArgOp->Asl.ParseOpcode == PARSEOP_DEREFOF))
340                 {
341                     break;
342                 }
344                 ThisNodeBtype = RequiredBtypes;
345                 break;
348             case ARGI_REFERENCE:            /* References */
349             case ARGI_INTEGER_REF:
350             case ARGI_OBJECT_REF:
351             case ARGI_DEVICE_REF:
353                 switch (ArgOp->Asl.ParseOpcode)
354                 {
355                 case PARSEOP_LOCAL0:
356                 case PARSEOP_LOCAL1:
357                 case PARSEOP_LOCAL2:
358                 case PARSEOP_LOCAL3:
359                 case PARSEOP_LOCAL4:
360                 case PARSEOP_LOCAL5:
361                 case PARSEOP_LOCAL6:
362                 case PARSEOP_LOCAL7:
364                     /* TBD: implement analysis of current value (type) of the lo
365                     /* For now, just treat any local as a typematch */
367                     /*ThisNodeBtype = RequiredBtypes;*/
368                     break;
370                 case PARSEOP_ARG0:
371                 case PARSEOP_ARG1:
372                 case PARSEOP_ARG2:
373                 case PARSEOP_ARG3:
374                 case PARSEOP_ARG4:
375                 case PARSEOP_ARG5:
376                 case PARSEOP_ARG6:
378                     /* Hard to analyze argument types, sow we won't */
379                     /* For now, just treat any arg as a typematch */
381                     /* ThisNodeBtype = RequiredBtypes; */
382                     break;
384                 case PARSEOP_DEBUG:
386                     break;
388                 case PARSEOP_REFOF:
389                 case PARSEOP_INDEX:
390                 default:
```

```
392                     break;

394                 }
395                 break;

397             case ARGI_INTEGER:
398             default:

400                 break;
401             }


404             CommonBtypes = ThisNodeBtype & RequiredBtypes;

406             if (ArgOp->Asl.ParseOpcode == PARSEOP_METHODCALL)
407             {
408                 if (AnIsInternalMethod (ArgOp))
409                 {
410                     return (AE_OK);
411                 }

413                 /* Check a method call for a valid return value */

415                 AnCheckMethodReturnValue (Op, OpInfo, ArgOp,
416                     RequiredBtypes, ThisNodeBtype);
417             }

419             /*
420              * Now check if the actual type(s) match at least one
421              * bit to the required type
422              */
423             else if (!CommonBtypes)
424             {
425                 /* No match -- this is a type mismatch error */

427                 AnFormatBtype (StringBuffer, ThisNodeBtype);
428                 AnFormatBtype (StringBuffer2, RequiredBtypes);

430                 sprintf (MsgBuffer, "[%s] found, %s operator requires [%s]",
431                             StringBuffer, OpInfo->Name, StringBuffer2);

433                 AslError (ASL_ERROR, ASL_MSG_INVALID_TYPE, ArgOp, MsgBuffer);
434             }

436         NextArgument:
437             ArgOp = ArgOp->Asl.Next;
438             INCREMENT_ARG_LIST (RuntimeArgTypes2);
439         }
440         break;

442     default:

444         break;
445     }

447     return (AE_OK);
448 }


451 /*******************************************************************************
452  *
453  * FUNCTION:    AnOtherSemanticAnalysisWalkBegin
454  *
455  * PARAMETERS:  ASL_WALK_CALLBACK
456  *
```



```
457  * RETURN:      Status
458  *
459  * DESCRIPTION: Descending callback for the analysis walk. Checks for
460  *              miscellaneous issues in the code.
461  *
462  ******************************************************************************/

464 ACPI_STATUS
465 AnOtherSemanticAnalysisWalkBegin (
466     ACPI_PARSE_OBJECT       *Op,
467     UINT32                  Level,
468     void                    *Context)
469 {
470     ACPI_PARSE_OBJECT       *ArgNode;
471     ACPI_PARSE_OBJECT       *PrevArgNode = NULL;
472     const ACPI_OPCODE_INFO  *OpInfo;
473     ACPI_NAMESPACE_NODE     *Node;


476     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);

478     /*
479      * Determine if an execution class operator actually does something by
480      * checking if it has a target and/or the function return value is used.
481      * (Target is optional, so a standalone statement can actually do nothing.)
482      */
483     if ((OpInfo->Class == AML_CLASS_EXECUTE) &&
484         (OpInfo->Flags & AML_HAS_RETVAL) &&
485         (!AnIsResultUsed (Op)))
486     {
487         if (OpInfo->Flags & AML_HAS_TARGET)
488         {
489             /*
490              * Find the target node, it is always the last child. If the traget
491              * is not specified in the ASL, a default node of type Zero was
492              * created by the parser.
493              */
494             ArgNode = Op->Asl.Child;
495             while (ArgNode->Asl.Next)
496             {
497                 PrevArgNode = ArgNode;
498                 ArgNode = ArgNode->Asl.Next;
499             }

501             /* Divide() is the only weird case, it has two targets */

503             if (Op->Asl.AmlOpcode == AML_DIVIDE_OP)
504             {
505                 if ((ArgNode->Asl.ParseOpcode == PARSEOP_ZERO) &&
506                     (PrevArgNode) &&
507                     (PrevArgNode->Asl.ParseOpcode == PARSEOP_ZERO))
508                 {
509                     AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
510                         Op, Op->Asl.ExternalName);
511                 }
512             }
513             else if (ArgNode->Asl.ParseOpcode == PARSEOP_ZERO)
514             {
515                 AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
516                     Op, Op->Asl.ExternalName);
517             }
518         }
519         else
520         {
521             /*
522              * Has no target and the result is not used. Only a couple opcodes
```

```
523              * can have this combination.
524              */
525             switch (Op->Asl.ParseOpcode)
526             {
527             case PARSEOP_ACQUIRE:
528             case PARSEOP_WAIT:
529             case PARSEOP_LOADTABLE:

531                 break;

533             default:

535                 AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
536                     Op, Op->Asl.ExternalName);
537                 break;
538             }
539         }
540     }


543     /*
544      * Semantic checks for individual ASL operators
545      */
546     switch (Op->Asl.ParseOpcode)
547     {
548     case PARSEOP_ACQUIRE:
549     case PARSEOP_WAIT:
550         /*
551          * Emit a warning if the timeout parameter for these operators is not
552          * ACPI_WAIT_FOREVER, and the result value from the operator is not
553          * checked, meaning that a timeout could happen, but the code
554          * would not know about it.
555          */

557         /* First child is the namepath, 2nd child is timeout */

559         ArgNode = Op->Asl.Child;
560         ArgNode = ArgNode->Asl.Next;

562         /*
563          * Check for the WAIT_FOREVER case - defined by the ACPI spec to be
564          * 0xFFFF or greater
565          */
566         if (((ArgNode->Asl.ParseOpcode == PARSEOP_WORDCONST) ||
567              (ArgNode->Asl.ParseOpcode == PARSEOP_INTEGER))  &&
568              (ArgNode->Asl.Value.Integer >= (UINT64) ACPI_WAIT_FOREVER))
569         {
570             break;
571         }

573         /*
574          * The operation could timeout. If the return value is not used
575          * (indicates timeout occurred), issue a warning
576          */
577         if (!AnIsResultUsed (Op))
578         {
579             AslError (ASL_WARNING, ASL_MSG_TIMEOUT, ArgNode,
580                 Op->Asl.ExternalName);
581         }
582         break;

584     case PARSEOP_CREATEFIELD:
585         /*
586          * Check for a zero Length (NumBits) operand. NumBits is the 3rd operand
587          */
588         ArgNode = Op->Asl.Child;
```



```
589         ArgNode = ArgNode->Asl.Next;
590         ArgNode = ArgNode->Asl.Next;

592         if ((ArgNode->Asl.ParseOpcode == PARSEOP_ZERO) ||
593            ((ArgNode->Asl.ParseOpcode == PARSEOP_INTEGER) &&
594             (ArgNode->Asl.Value.Integer == 0)))
595         {
596             AslError (ASL_ERROR, ASL_MSG_NON_ZERO, ArgNode, NULL);
597         }
598         break;

600     case PARSEOP_CONNECTION:
601         /*
602          * Ensure that the referenced operation region has the correct SPACE_ID.
603          * From the grammar/parser, we know the parent is a FIELD definition.
604          */
605         ArgNode = Op->Asl.Parent;       /* Field definition */
606         ArgNode = ArgNode->Asl.Child;   /* First child is the OpRegion Name */
607         Node = ArgNode->Asl.Node;       /* OpRegion namespace node */

609         ArgNode = Node->Op;             /* OpRegion definition */
610         ArgNode = ArgNode->Asl.Child;   /* First child is the OpRegion Name */
611         ArgNode = ArgNode->Asl.Next;    /* Next peer is the SPACE_ID (what we wa

613         /*
614          * The Connection() operator is only valid for the following operation
615          * region SpaceIds: GeneralPurposeIo and GenericSerialBus.
616          */
617         if ((ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GPIO) &&
618             (ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GSBUS))
619         {
620             AslError (ASL_ERROR, ASL_MSG_CONNECTION_INVALID, Op, NULL);
621         }
622         break;

624     case PARSEOP_FIELD:
625         /*
626          * Ensure that fields for GeneralPurposeIo and GenericSerialBus
627          * contain at least one Connection() operator
628          */
629         ArgNode = Op->Asl.Child;        /* 1st child is the OpRegion Name */
630         Node = ArgNode->Asl.Node;       /* OpRegion namespace node */
631         if (!Node)
632         {
633             break;
634         }

636         ArgNode = Node->Op;             /* OpRegion definition */
637         ArgNode = ArgNode->Asl.Child;   /* First child is the OpRegion Name */
638         ArgNode = ArgNode->Asl.Next;    /* Next peer is the SPACE_ID (what we wa

640         /* We are only interested in GeneralPurposeIo and GenericSerialBus */

642         if ((ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GPIO) &&
643             (ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GSBUS))
644         {
645             break;
646         }

648         ArgNode = Op->Asl.Child;        /* 1st child is the OpRegion Name */
649         ArgNode = ArgNode->Asl.Next;    /* AccessType */
650         ArgNode = ArgNode->Asl.Next;    /* LockRule */
651         ArgNode = ArgNode->Asl.Next;    /* UpdateRule */
652         ArgNode = ArgNode->Asl.Next;    /* Start of FieldUnitList */

654         /* Walk the FieldUnitList */
```

```
656         while (ArgNode)
657         {
658             if (ArgNode->Asl.ParseOpcode == PARSEOP_CONNECTION)
659             {
660                 break;
661             }
662             else if (ArgNode->Asl.ParseOpcode == PARSEOP_NAMESEG)
663             {
664                 AslError (ASL_ERROR, ASL_MSG_CONNECTION_MISSING, ArgNode, NULL);
665                 break;
666             }

668             ArgNode = ArgNode->Asl.Next;
669         }
670         break;

672     default:

674         break;
675     }

677     return (AE_OK);
678 }
```

```
**********************************************************
   27654 Thu Dec 26 13:48:36 2013
new/usr/src/common/acpica/compiler/aslxref.c
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aslxref - Namespace cross-reference
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "aslcompiler.h"
#include "aslcompiler.y.h"
#include "acparser.h"
#include "amlcode.h"
#include "acnamesp.h"
#include "acdispat.h"


#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("aslxref")

/* Local prototypes */

static ACPI_STATUS
XfNamespaceLocateBegin (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static ACPI_STATUS
XfNamespaceLocateEnd (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context);

static BOOLEAN
XfObjectExists (
    char                    *Name);

static ACPI_STATUS
XfCompareOneNamespaceObject (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

static void
XfCheckFieldRange (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  RegionBitLength,
    UINT32                  FieldBitOffset,
    UINT32                  FieldBitLength,
    UINT32                  AccessBitWidth);


/*******************************************************************************
 *
 * FUNCTION:    XfCrossReferenceNamespace
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform a cross reference check of the parse tree against the
 *              namespace. Every named referenced within the parse tree
 *              should be get resolved with a namespace lookup. If not, the
 *              original reference in the ASL code is invalid -- i.e., refers
 *              to a non-existent object.
 *
 * NOTE:  The ASL "External" operator causes the name to be inserted into the
 *        namespace so that references to the external name will be resolved
 *        correctly here.
 *
 ******************************************************************************/

ACPI_STATUS
XfCrossReferenceNamespace (
    void)
{
    ACPI_WALK_STATE         *WalkState;


    DbgPrint (ASL_DEBUG_OUTPUT, "\nCross referencing namespace\n\n");

    /*
     * Create a new walk state for use when looking up names
     * within the namespace (Passed as context to the callbacks)
     */
    WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
    if (!WalkState)
    {
        return (AE_NO_MEMORY);
```

```
127     }
129     /* Walk the entire parse tree */
131     TrWalkParseTree (RootNode, ASL_WALK_VISIT_TWICE, XfNamespaceLocateBegin,
132                         XfNamespaceLocateEnd, WalkState);
133     return (AE_OK);
134 }
137 /*******************************************************************************
138  *
139  * FUNCTION:    XfObjectExists
140  *
141  * PARAMETERS:  Name            - 4 char ACPI name
142  *
143  * RETURN:      TRUE if name exists in namespace
144  *
145  * DESCRIPTION: Walk the namespace to find an object
146  *
147  ******************************************************************************/
149 static BOOLEAN
150 XfObjectExists (
151     char                    *Name)
152 {
153     ACPI_STATUS             Status;
156     /* Walk entire namespace from the supplied root */
158     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
159                 ACPI_UINT32_MAX, FALSE, XfCompareOneNamespaceObject, NULL,
160                 Name, NULL);
161     if (Status == AE_CTRL_TRUE)
162     {
163         /* At least one instance of the name was found */
165         return (TRUE);
166     }
168     return (FALSE);
169 }
172 /*******************************************************************************
173  *
174  * FUNCTION:    XfCompareOneNamespaceObject
175  *
176  * PARAMETERS:  ACPI_WALK_CALLBACK
177  *
178  * RETURN:      Status
179  *
180  * DESCRIPTION: Compare name of one object.
181  *
182  ******************************************************************************/
184 static ACPI_STATUS
185 XfCompareOneNamespaceObject (
186     ACPI_HANDLE             ObjHandle,
187     UINT32                  Level,
188     void                    *Context,
189     void                    **ReturnValue)
190 {
191     ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
```



```
194     /* Simply check the name */
196     if (*((UINT32 *) (Context)) == Node->Name.Integer)
197     {
198         /* Abort walk if we found one instance */
200         return (AE_CTRL_TRUE);
201     }
203     return (AE_OK);
204 }
207 /*******************************************************************************
208  *
209  * FUNCTION:    XfCheckFieldRange
210  *
211  * PARAMETERS:  RegionBitLength     - Length of entire parent region
212  *              FieldBitOffset      - Start of the field unit (within region)
213  *              FieldBitLength      - Entire length of field unit
214  *              AccessBitWidth      - Access width of the field unit
215  *
216  * RETURN:      None
217  *
218  * DESCRIPTION: Check one field unit to make sure it fits in the parent
219  *              op region.
220  *
221  * Note: AccessBitWidth must be either 8,16,32, or 64
222  *
223  ******************************************************************************/
225 static void
226 XfCheckFieldRange (
227     ACPI_PARSE_OBJECT       *Op,
228     UINT32                  RegionBitLength,
229     UINT32                  FieldBitOffset,
230     UINT32                  FieldBitLength,
231     UINT32                  AccessBitWidth)
232 {
233     UINT32                  FieldEndBitOffset;
236     /*
237      * Check each field unit against the region size. The entire
238      * field unit (start offset plus length) must fit within the
239      * region.
240      */
241     FieldEndBitOffset = FieldBitOffset + FieldBitLength;
243     if (FieldEndBitOffset > RegionBitLength)
244     {
245         /* Field definition itself is beyond the end-of-region */
247         AslError (ASL_ERROR, ASL_MSG_FIELD_UNIT_OFFSET, Op, NULL);
248         return;
249     }
251     /*
252      * Now check that the field plus AccessWidth doesn't go beyond
253      * the end-of-region. Assumes AccessBitWidth is a power of 2
254      */
255     FieldEndBitOffset = ACPI_ROUND_UP (FieldEndBitOffset, AccessBitWidth);
257     if (FieldEndBitOffset > RegionBitLength)
258     {
```

```
259         /* Field definition combined with the access is beyond EOR */
260
261         AslError (ASL_ERROR, ASL_MSG_FIELD_UNIT_ACCESS_WIDTH, Op, NULL);
262     }
263 }
264
265 /*******************************************************************************
266  *
267  * FUNCTION:    XfNamespaceLocateBegin
268  *
269  * PARAMETERS:  ASL_WALK_CALLBACK
270  *
271  * RETURN:      Status
272  *
273  * DESCRIPTION: Descending callback used during cross-reference. For named
274  *              object references, attempt to locate the name in the
275  *              namespace.
276  *
277  * NOTE: ASL references to named fields within resource descriptors are
278  *       resolved to integer values here. Therefore, this step is an
279  *       important part of the code generation. We don't know that the
280  *       name refers to a resource descriptor until now.
281  *
282  ******************************************************************************/
283
284 static ACPI_STATUS
285 XfNamespaceLocateBegin (
286     ACPI_PARSE_OBJECT       *Op,
287     UINT32                  Level,
288     void                    *Context)
289 {
290     ACPI_WALK_STATE         *WalkState = (ACPI_WALK_STATE *) Context;
291     ACPI_NAMESPACE_NODE     *Node;
292     ACPI_STATUS             Status;
293     ACPI_OBJECT_TYPE        ObjectType;
294     char                    *Path;
295     UINT8                   PassedArgs;
296     ACPI_PARSE_OBJECT       *NextOp;
297     ACPI_PARSE_OBJECT       *OwningOp;
298     ACPI_PARSE_OBJECT       *SpaceIdOp;
299     UINT32                  MinimumLength;
300     UINT32                  Offset;
301     UINT32                  FieldBitLength;
302     UINT32                  TagBitLength;
303     UINT8                   Message = 0;
304     const ACPI_OPCODE_INFO  *OpInfo;
305     UINT32                  Flags;
306
307
308     ACPI_FUNCTION_TRACE_PTR (XfNamespaceLocateBegin, Op);
309
310     /*
311      * If this node is the actual declaration of a name
312      * [such as the XXXX name in "Method (XXXX)"],
313      * we are not interested in it here. We only care about names that are
314      * references to other objects within the namespace and the parent objects
315      * of name declarations
316      */
317     if (Op->Asl.CompileFlags & NODE_IS_NAME_DECLARATION)
318     {
319         return_ACPI_STATUS (AE_OK);
320     }
321
322     /* We are only interested in opcodes that have an associated name */
323
324     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
```



```
325
326     if ((!(OpInfo->Flags & AML_NAMED)) &&
327         (!(OpInfo->Flags & AML_CREATE)) &&
328         (Op->Asl.ParseOpcode != PARSEOP_NAMESTRING) &&
329         (Op->Asl.ParseOpcode != PARSEOP_NAMESEG)    &&
330         (Op->Asl.ParseOpcode != PARSEOP_METHODCALL))
331     {
332         return_ACPI_STATUS (AE_OK);
333     }
334
335     /*
336      * One special case: CondRefOf operator - we don't care if the name exists
337      * or not at this point, just ignore it, the point of the operator is to
338      * determine if the name exists at runtime.
339      */
340     if ((Op->Asl.Parent) &&
341         (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_CONDREFOF))
342     {
343         return_ACPI_STATUS (AE_OK);
344     }
345
346     /*
347      * We must enable the "search-to-root" for single NameSegs, but
348      * we have to be very careful about opening up scopes
349      */
350     Flags = ACPI_NS_SEARCH_PARENT;
351     if ((Op->Asl.ParseOpcode == PARSEOP_NAMESTRING) ||
352         (Op->Asl.ParseOpcode == PARSEOP_NAMESEG)    ||
353         (Op->Asl.ParseOpcode == PARSEOP_METHODCALL))
354     {
355         /*
356          * These are name references, do not push the scope stack
357          * for them.
358          */
359         Flags |= ACPI_NS_DONT_OPEN_SCOPE;
360     }
361
362     /* Get the NamePath from the appropriate place */
363
364     if (OpInfo->Flags & AML_NAMED)
365     {
366         /* For nearly all NAMED operators, the name reference is the first child
367
368         Path = Op->Asl.Child->Asl.Value.String;
369         if (Op->Asl.AmlOpcode == AML_ALIAS_OP)
370         {
371             /*
372              * ALIAS is the only oddball opcode, the name declaration
373              * (alias name) is the second operand
374              */
375             Path = Op->Asl.Child->Asl.Next->Asl.Value.String;
376         }
377     }
378     else if (OpInfo->Flags & AML_CREATE)
379     {
380         /* Name must appear as the last parameter */
381
382         NextOp = Op->Asl.Child;
383         while (!(NextOp->Asl.CompileFlags & NODE_IS_NAME_DECLARATION))
384         {
385             NextOp = NextOp->Asl.Next;
386         }
387         Path = NextOp->Asl.Value.String;
388     }
389     else
390     {
```

```
391         Path = Op->Asl.Value.String;
392     }
393
394     ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
395     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
396         "Type=%s\n", AcpiUtGetTypeName (ObjectType)));
397
398     /*
399      * Lookup the name in the namespace. Name must exist at this point, or it
400      * is an invalid reference.
401      *
402      * The namespace is also used as a lookup table for references to resource
403      * descriptors and the fields within them.
404      */
405     Gbl_NsLookupCount++;
406
407     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
408                 ACPI_IMODE_EXECUTE, Flags, WalkState, &(Node));
409     if (ACPI_FAILURE (Status))
410     {
411         if (Status == AE_NOT_FOUND)
412         {
413             /*
414              * We didn't find the name reference by path -- we can qualify this
415              * a little better before we print an error message
416              */
417             if (strlen (Path) == ACPI_NAME_SIZE)
418             {
419                 /* A simple, one-segment ACPI name */
420
421                 if (XfObjectExists (Path))
422                 {
423                     /*
424                      * There exists such a name, but we couldn't get to it
425                      * from this scope
426                      */
427                     AslError (ASL_ERROR, ASL_MSG_NOT_REACHABLE, Op,
428                         Op->Asl.ExternalName);
429                 }
430                 else
431                 {
432                     /* The name doesn't exist, period */
433
434                     AslError (ASL_ERROR, ASL_MSG_NOT_EXIST,
435                         Op, Op->Asl.ExternalName);
436                 }
437             }
438             else
439             {
440                 /* Check for a fully qualified path */
441
442                 if (Path[0] == AML_ROOT_PREFIX)
443                 {
444                     /* Gave full path, the object does not exist */
445
446                     AslError (ASL_ERROR, ASL_MSG_NOT_EXIST, Op,
447                         Op->Asl.ExternalName);
448                 }
449                 else
450                 {
451                     /*
452                      * We can't tell whether it doesn't exist or just
453                      * can't be reached.
454                      */
455                     AslError (ASL_ERROR, ASL_MSG_NOT_FOUND, Op,
456                         Op->Asl.ExternalName);
```



```
457                 }
458             }
459
460             Status = AE_OK;
461         }
462
463         return_ACPI_STATUS (Status);
464     }
465
466     /* Check for a reference vs. name declaration */
467
468     if (!(OpInfo->Flags & AML_NAMED) &&
469         !(OpInfo->Flags & AML_CREATE))
470     {
471         /* This node has been referenced, mark it for reference check */
472
473         Node->Flags |= ANOBJ_IS_REFERENCED;
474     }
475
476     /* Attempt to optimize the NamePath */
477
478     OptOptimizeNamePath (Op, OpInfo->Flags, WalkState, Path, Node);
479
480     /*
481      * 1) Dereference an alias (A name reference that is an alias)
482      *    Aliases are not nested, the alias always points to the final object
483      */
484     if ((Op->Asl.ParseOpcode != PARSEOP_ALIAS) &&
485         (Node->Type == ACPI_TYPE_LOCAL_ALIAS))
486     {
487         /* This node points back to the original PARSEOP_ALIAS */
488
489         NextOp = Node->Op;
490
491         /* The first child is the alias target op */
492
493         NextOp = NextOp->Asl.Child;
494
495         /* That in turn points back to original target alias node */
496
497         if (NextOp->Asl.Node)
498         {
499             Node = NextOp->Asl.Node;
500         }
501
502         /* Else - forward reference to alias, will be resolved later */
503     }
504
505     /* 2) Check for a reference to a resource descriptor */
506
507     if ((Node->Type == ACPI_TYPE_LOCAL_RESOURCE_FIELD) ||
508         (Node->Type == ACPI_TYPE_LOCAL_RESOURCE))
509     {
510         /*
511          * This was a reference to a field within a resource descriptor.
512          * Extract the associated field offset (either a bit or byte
513          * offset depending on the field type) and change the named
514          * reference into an integer for AML code generation
515          */
516         Offset = Node->Value;
517         TagBitLength = Node->Length;
518
519         /*
520          * If a field is being created, generate the length (in bits) of
521          * the field. Note: Opcodes other than CreateXxxField and Index
522          * can come through here. For other opcodes, we just need to
```

```
523          * convert the resource tag reference to an integer offset.
524          */
525         switch (Op->Asl.Parent->Asl.AmlOpcode)
526         {
527         case AML_CREATE_FIELD_OP: /* Variable "Length" field, in bits */
528             /*
529              * We know the length operand is an integer constant because
530              * we know that it contains a reference to a resource
531              * descriptor tag.
532              */
533             FieldBitLength = (UINT32) Op->Asl.Next->Asl.Value.Integer;
534             break;

536         case AML_CREATE_BIT_FIELD_OP:

538             FieldBitLength = 1;
539             break;

541         case AML_CREATE_BYTE_FIELD_OP:
542         case AML_INDEX_OP:

544             FieldBitLength = 8;
545             break;

547         case AML_CREATE_WORD_FIELD_OP:

549             FieldBitLength = 16;
550             break;

552         case AML_CREATE_DWORD_FIELD_OP:

554             FieldBitLength = 32;
555             break;

557         case AML_CREATE_QWORD_FIELD_OP:

559             FieldBitLength = 64;
560             break;

562         default:

564             FieldBitLength = 0;
565             break;
566         }

568         /* Check the field length against the length of the resource tag */

570         if (FieldBitLength)
571         {
572             if (TagBitLength < FieldBitLength)
573             {
574                 Message = ASL_MSG_TAG_SMALLER;
575             }
576             else if (TagBitLength > FieldBitLength)
577             {
578                 Message = ASL_MSG_TAG_LARGER;
579             }

581             if (Message)
582             {
583                 sprintf (MsgBuffer, "Size mismatch, Tag: %u bit%s, Field: %u bit
584                     TagBitLength, (TagBitLength > 1) ? "s" : "",
585                     FieldBitLength, (FieldBitLength > 1) ? "s" : "");

587                 AslError (ASL_WARNING, Message, Op, MsgBuffer);
588             }
```



```
589         }

591         /* Convert the BitOffset to a ByteOffset for certain opcodes */

593         switch (Op->Asl.Parent->Asl.AmlOpcode)
594         {
595         case AML_CREATE_BYTE_FIELD_OP:
596         case AML_CREATE_WORD_FIELD_OP:
597         case AML_CREATE_DWORD_FIELD_OP:
598         case AML_CREATE_QWORD_FIELD_OP:
599         case AML_INDEX_OP:

601             Offset = ACPI_DIV_8 (Offset);
602             break;

604         default:

606             break;
607         }

609         /* Now convert this node to an integer whose value is the field offset *

611         Op->Asl.AmlLength = 0;
612         Op->Asl.ParseOpcode = PARSEOP_INTEGER;
613         Op->Asl.Value.Integer = (UINT64) Offset;
614         Op->Asl.CompileFlags |= NODE_IS_RESOURCE_FIELD;

616         OpcGenerateAmlOpcode (Op);
617     }

619     /* 3) Check for a method invocation */

621     else if ((((Op->Asl.ParseOpcode == PARSEOP_NAMESTRING) || (Op->Asl.ParseOpco
622                 (Node->Type == ACPI_TYPE_METHOD) &&
623                 (Op->Asl.Parent) &&
624                 (Op->Asl.Parent->Asl.ParseOpcode != PARSEOP_METHOD))   ||

626                 (Op->Asl.ParseOpcode == PARSEOP_METHODCALL))
627     {
628         /*
629          * A reference to a method within one of these opcodes is not an
630          * invocation of the method, it is simply a reference to the method.
631          */
632         if ((Op->Asl.Parent) &&
633            ((Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_REFOF)      ||
634             (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_DEREFOF)    ||
635             (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_OBJECTTYPE)))
636         {
637             return_ACPI_STATUS (AE_OK);
638         }
639         /*
640          * There are two types of method invocation:
641          * 1) Invocation with arguments -- the parser recognizes this
642          *    as a METHODCALL.
643          * 2) Invocation with no arguments --the parser cannot determine that
644          *    this is a method invocation, therefore we have to figure it out
645          *    here.
646          */
647         if (Node->Type != ACPI_TYPE_METHOD)
648         {
649             sprintf (MsgBuffer, "%s is a %s",
650                     Op->Asl.ExternalName, AcpiUtGetTypeName (Node->Type));

652             AslError (ASL_ERROR, ASL_MSG_NOT_METHOD, Op, MsgBuffer);
653             return_ACPI_STATUS (AE_OK);
654         }
```

```
656         /* Save the method node in the caller's op */

658         Op->Asl.Node = Node;
659         if (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_CONDREFOF)
660         {
661             return_ACPI_STATUS (AE_OK);
662         }

664         /*
665          * This is a method invocation, with or without arguments.
666          * Count the number of arguments, each appears as a child
667          * under the parent node
668          */
669         Op->Asl.ParseOpcode = PARSEOP_METHODCALL;
670         UtSetParseOpName (Op);

672         PassedArgs = 0;
673         NextOp     = Op->Asl.Child;

675         while (NextOp)
676         {
677             PassedArgs++;
678             NextOp = NextOp->Asl.Next;
679         }

681         if (Node->Value != ASL_EXTERNAL_METHOD)
682         {
683             /*
684              * Check the parsed arguments with the number expected by the
685              * method declaration itself
686              */
687             if (PassedArgs != Node->Value)
688             {
689                 sprintf (MsgBuffer, "%s requires %u", Op->Asl.ExternalName,
690                             Node->Value);

692                 if (PassedArgs < Node->Value)
693                 {
694                     AslError (ASL_ERROR, ASL_MSG_ARG_COUNT_LO, Op, MsgBuffer);
695                 }
696                 else
697                 {
698                     AslError (ASL_ERROR, ASL_MSG_ARG_COUNT_HI, Op, MsgBuffer);
699                 }
700             }
701         }
702     }

704     /* 4) Check for an ASL Field definition */

706     else if ((Op->Asl.Parent) &&
707             ((Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_FIELD)     ||
708              (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_BANKFIELD)))
709     {
710         /*
711          * Offset checking for fields. If the parent operation region has a
712          * constant length (known at compile time), we can check fields
713          * defined in that region against the region length. This will catch
714          * fields and field units that cannot possibly fit within the region.
715          *
716          * Note: Index fields do not directly reference an operation region,
717          * thus they are not included in this check.
718          */
719         if (Op == Op->Asl.Parent->Asl.Child)
720         {
```



```
721             /*
722              * This is the first child of the field node, which is
723              * the name of the region. Get the parse node for the
724              * region -- which contains the length of the region.
725              */
726             OwningOp = Node->Op;
727             Op->Asl.Parent->Asl.ExtraValue =
728                 ACPI_MUL_8 ((UINT32) OwningOp->Asl.Value.Integer);

730             /* Examine the field access width */

732             switch ((UINT8) Op->Asl.Parent->Asl.Value.Integer)
733             {
734             case AML_FIELD_ACCESS_ANY:
735             case AML_FIELD_ACCESS_BYTE:
736             case AML_FIELD_ACCESS_BUFFER:
737             default:

739                 MinimumLength = 1;
740                 break;

742             case AML_FIELD_ACCESS_WORD:

744                 MinimumLength = 2;
745                 break;

747             case AML_FIELD_ACCESS_DWORD:

749                 MinimumLength = 4;
750                 break;

752             case AML_FIELD_ACCESS_QWORD:

754                 MinimumLength = 8;
755                 break;
756             }

758             /*
759              * Is the region at least as big as the access width?
760              * Note: DataTableRegions have 0 length
761              */
762             if (((UINT32) OwningOp->Asl.Value.Integer) &&
763                 ((UINT32) OwningOp->Asl.Value.Integer < MinimumLength))
764             {
765                 AslError (ASL_ERROR, ASL_MSG_FIELD_ACCESS_WIDTH, Op, NULL);
766             }

768             /*
769              * Check EC/CMOS/SMBUS fields to make sure that the correct
770              * access type is used (BYTE for EC/CMOS, BUFFER for SMBUS)
771              */
772             SpaceIdOp = OwningOp->Asl.Child->Asl.Next;
773             switch ((UINT32) SpaceIdOp->Asl.Value.Integer)
774             {
775             case ACPI_ADR_SPACE_EC:
776             case ACPI_ADR_SPACE_CMOS:
777             case ACPI_ADR_SPACE_GPIO:

779                 if ((UINT8) Op->Asl.Parent->Asl.Value.Integer != AML_FIELD_ACCES
780                 {
781                     AslError (ASL_ERROR, ASL_MSG_REGION_BYTE_ACCESS, Op, NULL);
782                 }
783                 break;

785             case ACPI_ADR_SPACE_SMBUS:
786             case ACPI_ADR_SPACE_IPMI:
```

```
787             case ACPI_ADR_SPACE_GSBUS:

789                 if ((UINT8) Op->Asl.Parent->Asl.Value.Integer != AML_FIELD_ACCES
790                 {
791                     AslError (ASL_ERROR, ASL_MSG_REGION_BUFFER_ACCESS, Op, NULL)
792                 }
793                 break;

795             default:

797                 /* Nothing to do for other address spaces */

799                 break;
800             }
801         }
802         else
803         {
804             /*
805              * This is one element of the field list. Check to make sure
806              * that it does not go beyond the end of the parent operation region
807              *
808              * In the code below:
809              *    Op->Asl.Parent->Asl.ExtraValue      - Region Length (bits)
810              *    Op->Asl.ExtraValue                  - Field start offset (bits
811              *    Op->Asl.Child->Asl.Value.Integer32  - Field length (bits)
812              *    Op->Asl.Child->Asl.ExtraValue       - Field access width (bits
813              */
814             if (Op->Asl.Parent->Asl.ExtraValue && Op->Asl.Child)
815             {
816                 XfCheckFieldRange (Op,
817                             Op->Asl.Parent->Asl.ExtraValue,
818                             Op->Asl.ExtraValue,
819                             (UINT32) Op->Asl.Child->Asl.Value.Integer,
820                             Op->Asl.Child->Asl.ExtraValue);
821             }
822         }
823     }

825     Op->Asl.Node = Node;
826     return_ACPI_STATUS (Status);
827 }


830 /*******************************************************************************
831  *
832  * FUNCTION:    XfNamespaceLocateEnd
833  *
834  * PARAMETERS:  ASL_WALK_CALLBACK
835  *
836  * RETURN:      Status
837  *
838  * DESCRIPTION: Ascending callback used during cross reference. We only
839  *              need to worry about scope management here.
840  *
841  ******************************************************************************/

843 static ACPI_STATUS
844 XfNamespaceLocateEnd (
845     ACPI_PARSE_OBJECT       *Op,
846     UINT32                  Level,
847     void                    *Context)
848 {
849     ACPI_WALK_STATE         *WalkState = (ACPI_WALK_STATE *) Context;
850     const ACPI_OPCODE_INFO  *OpInfo;
```



```
853     ACPI_FUNCTION_TRACE (XfNamespaceLocateEnd);


856     /* We are only interested in opcodes that have an associated name */

858     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
859     if (!(OpInfo->Flags & AML_NAMED))
860     {
861         return_ACPI_STATUS (AE_OK);
862     }

864     /* Not interested in name references, we did not open a scope for them */

866     if ((Op->Asl.ParseOpcode == PARSEOP_NAMESTRING) ||
867         (Op->Asl.ParseOpcode == PARSEOP_NAMESEG)    ||
868         (Op->Asl.ParseOpcode == PARSEOP_METHODCALL))
869     {
870         return_ACPI_STATUS (AE_OK);
871     }

873     /* Pop the scope stack if necessary */

875     if (AcpiNsOpensScope (AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode)))
876     {

878         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
879             "%s: Popping scope for Op %p\n",
880             AcpiUtGetTypeName (OpInfo->ObjectType), Op));

882         (void) AcpiDsScopeStackPop (WalkState);
883     }

885     return_ACPI_STATUS (AE_OK);
886 }
```

```
**********************************************************
   16440 Thu Dec 26 13:48:36 2013
new/usr/src/common/acpica/compiler/dtcompile.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dtcompile.c - Front-end for data table compiler
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DTCOMPILE_C__
  45 #define _DECLARE_DT_GLOBALS

  47 #include "aslcompiler.h"
  48 #include "dtcompiler.h"

  50 #define _COMPONENT          DT_COMPILER
  51          ACPI_MODULE_NAME    ("dtcompile")

  53 static char                 VersionString[9];


  56 /* Local prototypes */

  58 static ACPI_STATUS
  59 DtInitialize (
```



```
  60     void);

  62 static ACPI_STATUS
  63 DtCompileDataTable (
  64     DT_FIELD                **Field);

  66 static void
  67 DtInsertCompilerIds (
  68     DT_FIELD                *FieldList);


  71 /******************************************************************************
  72  *
  73  * FUNCTION:    DtDoCompile
  74  *
  75  * PARAMETERS:  None
  76  *
  77  * RETURN:      Status
  78  *
  79  * DESCRIPTION: Main entry point for the data table compiler.
  80  *
  81  * Note: Assumes Gbl_Files[ASL_FILE_INPUT] is initialized and the file is
  82  *          open at seek offset zero.
  83  *
  84  *****************************************************************************/

  86 ACPI_STATUS
  87 DtDoCompile (
  88     void)
  89 {
  90     ACPI_STATUS             Status;
  91     UINT8                   Event;
  92     DT_FIELD                *FieldList;


  95     /* Initialize globals */

  97     Status = DtInitialize ();
  98     if (ACPI_FAILURE (Status))
  99     {
 100         printf ("Error during compiler initialization, 0x%X\n", Status);
 101         return (Status);
 102     }

 104     /* Preprocessor */

 106     Event = UtBeginEvent ("Preprocess input file");
 107     PrDoPreprocess ();
 108     UtEndEvent (Event);

 110     if (Gbl_PreprocessOnly)
 111     {
 112         return (AE_OK);
 113     }

 115     /*
 116      * Scan the input file (file is already open) and
 117      * build the parse tree
 118      */
 119     Event = UtBeginEvent ("Scan and parse input file");
 120     FieldList = DtScanFile (Gbl_Files[ASL_FILE_INPUT].Handle);
 121     UtEndEvent (Event);

 123     /* Did the parse tree get successfully constructed? */

 125     if (!FieldList)
```

```
126     {
127         /* TBD: temporary error message. Msgs should come from function above */
128
129         DtError (ASL_ERROR, ASL_MSG_SYNTAX, NULL,
130             "Input file does not appear to be an ASL or data table source file")
131
132         Status = AE_ERROR;
133         goto CleanupAndExit;
134     }
135
136     Event = UtBeginEvent ("Compile parse tree");
137
138     /*
139      * Compile the parse tree
140      */
141     Status = DtCompileDataTable (&FieldList);
142     UtEndEvent (Event);
143
144     DtFreeFieldList ();
145
146     if (ACPI_FAILURE (Status))
147     {
148         /* TBD: temporary error message. Msgs should come from function above */
149
150         DtError (ASL_ERROR, ASL_MSG_SYNTAX, NULL,
151             "Could not compile input file");
152
153         goto CleanupAndExit;
154     }
155
156     /* Create/open the binary output file */
157
158     Gbl_Files[ASL_FILE_AML_OUTPUT].Filename = NULL;
159     Status = FlOpenAmlOutputFile (Gbl_OutputFilenamePrefix);
160     if (ACPI_FAILURE (Status))
161     {
162         goto CleanupAndExit;
163     }
164
165     /* Write the binary, then the optional hex file */
166
167     DtOutputBinary (Gbl_RootTable);
168     HxDoHexOutput ();
169     DtWriteTableToListing ();
170
171 CleanupAndExit:
172
173     CmCleanupAndExit ();
174     return (Status);
175 }
176
177
178 /******************************************************************************
179  *
180  * FUNCTION:    DtInitialize
181  *
182  * PARAMETERS:  None
183  *
184  * RETURN:      Status
185  *
186  * DESCRIPTION: Initialize data table compiler globals. Enables multiple
187  *              compiles per invocation.
188  *
189  *****************************************************************************/
190
191 static ACPI_STATUS
```



```
192 DtInitialize (
193     void)
194 {
195     ACPI_STATUS             Status;
196
197
198     Status = AcpiOsInitialize ();
199     if (ACPI_FAILURE (Status))
200     {
201         return (Status);
202     }
203
204     Status = AcpiUtInitGlobals ();
205     if (ACPI_FAILURE (Status))
206     {
207         return (Status);
208     }
209
210     Gbl_FieldList = NULL;
211     Gbl_RootTable = NULL;
212     Gbl_SubtableStack = NULL;
213
214     sprintf (VersionString, "%X", (UINT32) ACPI_CA_VERSION);
215     return (AE_OK);
216 }
217
218
219 /******************************************************************************
220  *
221  * FUNCTION:    DtInsertCompilerIds
222  *
223  * PARAMETERS:  FieldList           - Current field list pointer
224  *
225  * RETURN:      None
226  *
227  * DESCRIPTION: Insert the IDs (Name, Version) of the current compiler into
228  *              the original ACPI table header.
229  *
230  *****************************************************************************/
231
232 static void
233 DtInsertCompilerIds (
234     DT_FIELD                *FieldList)
235 {
236     DT_FIELD                *Next;
237     UINT32                  i;
238
239
240     /*
241      * Don't insert current compiler ID if requested. Used for compiler
242      * debug/validation only.
243      */
244     if (Gbl_UseOriginalCompilerId)
245     {
246         return;
247     }
248
249     /* Walk to the Compiler fields at the end of the header */
250
251     Next = FieldList;
252     for (i = 0; i < 7; i++)
253     {
254         Next = Next->Next;
255     }
256
257     Next->Value = ASL_CREATOR_ID;
```

```
258     Next->Flags = DT_FIELD_NOT_ALLOCATED;

260     Next = Next->Next;
261     Next->Value = VersionString;
262     Next->Flags = DT_FIELD_NOT_ALLOCATED;
263 }


266 /******************************************************************************
267  *
268  * FUNCTION:    DtCompileDataTable
269  *
270  * PARAMETERS:  FieldList           - Current field list pointer
271  *
272  * RETURN:      Status
273  *
274  * DESCRIPTION: Entry point to compile one data table
275  *
276  *****************************************************************************/

278 static ACPI_STATUS
279 DtCompileDataTable (
280     DT_FIELD                **FieldList)
281 {
282     ACPI_DMTABLE_DATA       *TableData;
283     DT_SUBTABLE             *Subtable;
284     char                    *Signature;
285     ACPI_TABLE_HEADER       *AcpiTableHeader;
286     ACPI_STATUS             Status;
287     DT_FIELD                *RootField = *FieldList;


290     /* Verify that we at least have a table signature and save it */

292     Signature = DtGetFieldValue (*FieldList);
293     if (!Signature)
294     {
295         sprintf (MsgBuffer, "Expected \"%s\"", "Signature");
296         DtNameError (ASL_ERROR, ASL_MSG_INVALID_FIELD_NAME,
297             *FieldList, MsgBuffer);
298         return (AE_ERROR);
299     }

301     Gbl_Signature = UtLocalCalloc (ACPI_STRLEN (Signature) + 1);
302     strcpy (Gbl_Signature, Signature);

304     /*
305      * Handle tables that don't use the common ACPI table header structure.
306      * Currently, these are the FACS and RSDP. Also check for an OEMx table,
307      * these tables have user-defined contents.
308      */
309     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS))
310     {
311         Status = DtCompileFacs (FieldList);
312         if (ACPI_FAILURE (Status))
313         {
314             return (Status);
315         }

317         DtSetTableLength ();
318         return (Status);
319     }
320     else if (ACPI_VALIDATE_RSDP_SIG (Signature))
321     {
322         Status = DtCompileRsdp (FieldList);
323         return (Status);
```



```
324     }
325     else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_S3PT))
326     {
327         Status = DtCompileS3pt (FieldList);
328         if (ACPI_FAILURE (Status))
329         {
330             return (Status);
331         }

333         DtSetTableLength ();
334         return (Status);
335     }

337     /*
338      * All other tables must use the common ACPI table header. Insert the
339      * current iASL IDs (name, version), and compile the header now.
340      */
341     DtInsertCompilerIds (*FieldList);

343     Status = DtCompileTable (FieldList, AcpiDmTableInfoHeader,
344                 &Gbl_RootTable, TRUE);
345     if (ACPI_FAILURE (Status))
346     {
347         return (Status);
348     }

350     DtPushSubtable (Gbl_RootTable);

352     /* Validate the signature via the ACPI table list */

354     TableData = AcpiDmGetTableData (Signature);
355     if (!TableData || Gbl_CompileGeneric)
356     {
357         DtCompileGeneric ((void **) FieldList);
358         goto FinishHeader;
359     }

361     /* Dispatch to per-table compile */

363     if (TableData->CmTableHandler)
364     {
365         /* Complex table, has a handler */

367         Status = TableData->CmTableHandler ((void **) FieldList);
368         if (ACPI_FAILURE (Status))
369         {
370             return (Status);
371         }
372     }
373     else if (TableData->TableInfo)
374     {
375         /* Simple table, just walk the info table */

377         Subtable = NULL;
378         Status = DtCompileTable (FieldList, TableData->TableInfo,
379                     &Subtable, TRUE);
380         if (ACPI_FAILURE (Status))
381         {
382             return (Status);
383         }

385         DtInsertSubtable (Gbl_RootTable, Subtable);
386         DtPopSubtable ();
387     }
388     else
389     {
```

```
390         DtFatal (ASL_MSG_COMPILER_INTERNAL, *FieldList,
391             "Missing table dispatch info");
392         return (AE_ERROR);
393     }

395 FinishHeader:

397     /* Set the final table length and then the checksum */

399     DtSetTableLength ();
400     AcpiTableHeader = ACPI_CAST_PTR (
401         ACPI_TABLE_HEADER, Gbl_RootTable->Buffer);
402     DtSetTableChecksum (&AcpiTableHeader->Checksum);

404     DtDumpFieldList (RootField);
405     DtDumpSubtableList ();
406     return (AE_OK);
407 }


410 /******************************************************************************
411  *
412  * FUNCTION:    DtCompileTable
413  *
414  * PARAMETERS:  Field               - Current field list pointer
415  *              Info                - Info table for this ACPI table
416  *              RetSubtable         - Compile result of table
417  *              Required            - If this subtable must exist
418  *
419  * RETURN:      Status
420  *
421  * DESCRIPTION: Compile a subtable
422  *
423  *****************************************************************************/

425 ACPI_STATUS
426 DtCompileTable (
427     DT_FIELD                **Field,
428     ACPI_DMTABLE_INFO       *Info,
429     DT_SUBTABLE             **RetSubtable,
430     BOOLEAN                 Required)
431 {
432     DT_FIELD                *LocalField;
433     UINT32                  Length;
434     DT_SUBTABLE             *Subtable;
435     DT_SUBTABLE             *InlineSubtable;
436     UINT32                  FieldLength = 0;
437     UINT8                   FieldType;
438     UINT8                   *Buffer;
439     UINT8                   *FlagBuffer = NULL;
440     UINT32                  CurrentFlagByteOffset = 0;
441     ACPI_STATUS             Status;


444     if (!Field || !*Field)
445     {
446         return (AE_BAD_PARAMETER);
447     }

449     /* Ignore optional subtable if name does not match */

451     if ((Info->Flags & DT_OPTIONAL) &&
452         ACPI_STRCMP ((*Field)->Name, Info->Name))
453     {
454         *RetSubtable = NULL;
455         return (AE_OK);
```



```
456     }

458     Length = DtGetSubtableLength (*Field, Info);
459     if (Length == ASL_EOF)
460     {
461         return (AE_ERROR);
462     }

464     Subtable = UtLocalCalloc (sizeof (DT_SUBTABLE));

466     if (Length > 0)
467     {
468         Subtable->Buffer = UtLocalCalloc (Length);
469     }
470     Subtable->Length = Length;
471     Subtable->TotalLength = Length;
472     Buffer = Subtable->Buffer;

474     LocalField = *Field;

476     /*
477      * Main loop walks the info table for this ACPI table or subtable
478      */
479     for (; Info->Name; Info++)
480     {
481         if (Info->Opcode == ACPI_DMT_EXTRA_TEXT)
482         {
483             continue;
484         }

486         if (!LocalField)
487         {
488             sprintf (MsgBuffer, "Found NULL field - Field name \"%s\" needed",
489                 Info->Name);
490             DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, MsgBuffer);
491             Status = AE_BAD_DATA;
492             goto Error;
493         }

495         /* Maintain table offsets */

497         LocalField->TableOffset = Gbl_CurrentTableOffset;
498         FieldLength = DtGetFieldLength (LocalField, Info);
499         Gbl_CurrentTableOffset += FieldLength;

501         FieldType = DtGetFieldType (Info);
502         Gbl_InputFieldCount++;

504         switch (FieldType)
505         {
506         case DT_FIELD_TYPE_FLAGS_INTEGER:
507             /*
508              * Start of the definition of a flags field.
509              * This master flags integer starts at value zero, in preparation
510              * to compile and insert the flag fields from the individual bits
511              */
512             LocalField = LocalField->Next;
513             *Field = LocalField;

515             FlagBuffer = Buffer;
516             CurrentFlagByteOffset = Info->Offset;
517             break;

519         case DT_FIELD_TYPE_FLAG:

521             /* Individual Flag field, can be multiple bits */
```

```
523             if (FlagBuffer)
524             {
525                 /*
526                  * We must increment the FlagBuffer when we have crossed
527                  * into the next flags byte within the flags field
528                  * of type DT_FIELD_TYPE_FLAGS_INTEGER.
529                  */
530                 FlagBuffer += (Info->Offset - CurrentFlagByteOffset);
531                 CurrentFlagByteOffset = Info->Offset;
532
533                 DtCompileFlag (FlagBuffer, LocalField, Info);
534             }
535             else
536             {
537                 /* TBD - this is an internal error */
538             }
539
540             LocalField = LocalField->Next;
541             *Field = LocalField;
542             break;
543
544         case DT_FIELD_TYPE_INLINE_SUBTABLE:
545             /*
546              * Recursion (one level max): compile GAS (Generic Address)
547              * or Notify in-line subtable
548              */
549             *Field = LocalField;
550
551             if (Info->Opcode == ACPI_DMT_GAS)
552             {
553                 Status = DtCompileTable (Field, AcpiDmTableInfoGas,
554                     &InlineSubtable, TRUE);
555             }
556             else
557             {
558                 Status = DtCompileTable (Field, AcpiDmTableInfoHestNotify,
559                     &InlineSubtable, TRUE);
560             }
561
562             if (ACPI_FAILURE (Status))
563             {
564                 goto Error;
565             }
566
567             DtSetSubtableLength (InlineSubtable);
568
569             ACPI_MEMCPY (Buffer, InlineSubtable->Buffer, FieldLength);
570             ACPI_FREE (InlineSubtable->Buffer);
571             ACPI_FREE (InlineSubtable);
572             LocalField = *Field;
573             break;
574
575         case DT_FIELD_TYPE_LABEL:
576
577             DtWriteFieldToListing (Buffer, LocalField, 0);
578             LocalField = LocalField->Next;
579             break;
580
581         default:
582
583             /* Normal case for most field types (Integer, String, etc.) */
584
585             DtCompileOneField (Buffer, LocalField,
586                 FieldLength, FieldType, Info->Flags);
```



```
588             DtWriteFieldToListing (Buffer, LocalField, FieldLength);
589             LocalField = LocalField->Next;
590
591             if (Info->Flags & DT_LENGTH)
592             {
593                 /* Field is an Integer that will contain a subtable length */
594
595                 Subtable->LengthField = Buffer;
596                 Subtable->SizeOfLengthField = FieldLength;
597             }
598
599             break;
600         }
601
602         Buffer += FieldLength;
603     }
604
605     *Field = LocalField;
606     *RetSubtable = Subtable;
607     return (AE_OK);
608
609 Error:
610     ACPI_FREE (Subtable->Buffer);
611     ACPI_FREE (Subtable);
612     return (Status);
613 }
```

```
**********************************************************
   13639 Thu Dec 26 13:48:36 2013
new/usr/src/common/acpica/compiler/dtcompiler.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dtcompiler.h - header for data table compiler
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DTCOMPILER_H__

  46 #ifndef _DTCOMPILER
  47 #define _DTCOMPILER

  49 #include <stdio.h>
  50 #include "acdisasm.h"


  53 #undef DT_EXTERN

  55 #ifdef _DECLARE_DT_GLOBALS
  56 #define DT_EXTERN
  57 #define DT_INIT_GLOBAL(a,b)         (a)=(b)
  58 #else
  59 #define DT_EXTERN                   extern
```



```
  60 #define DT_INIT_GLOBAL(a,b)         (a)
  61 #endif


  64 /* Types for individual fields (one per input line) */

  66 #define DT_FIELD_TYPE_STRING            0
  67 #define DT_FIELD_TYPE_INTEGER           1
  68 #define DT_FIELD_TYPE_BUFFER            2
  69 #define DT_FIELD_TYPE_PCI_PATH          3
  70 #define DT_FIELD_TYPE_FLAG              4
  71 #define DT_FIELD_TYPE_FLAGS_INTEGER     5
  72 #define DT_FIELD_TYPE_INLINE_SUBTABLE   6
  73 #define DT_FIELD_TYPE_UUID              7
  74 #define DT_FIELD_TYPE_UNICODE           8
  75 #define DT_FIELD_TYPE_DEVICE_PATH       9
  76 #define DT_FIELD_TYPE_LABEL             10


  79 /*
  80  * Structure used for each individual field within an ACPI table
  81  */
  82 typedef struct dt_field
  83 {
  84     char                    *Name;      /* Field name (from name : value) */
  85     char                    *Value;     /* Field value (from name : value) */
  86     struct dt_field         *Next;      /* Next field */
  87     struct dt_field         *NextLabel; /* If field is a label, next label */
  88     UINT32                  Line;       /* Line number for this field */
  89     UINT32                  ByteOffset; /* Offset in source file for field */
  90     UINT32                  NameColumn; /* Start column for field name */
  91     UINT32                  Column;     /* Start column for field value */
  92     UINT32                  TableOffset;/* Binary offset within ACPI table */
  93     UINT8                   Flags;

  95 } DT_FIELD;

  97 /* Flags for above */

  99 #define DT_FIELD_NOT_ALLOCATED      1


 102 /*
 103  * Structure used for individual subtables within an ACPI table
 104  */
 105 typedef struct dt_subtable
 106 {
 107     struct dt_subtable      *Parent;
 108     struct dt_subtable      *Child;
 109     struct dt_subtable      *Peer;
 110     struct dt_subtable      *StackTop;
 111     UINT8                   *Buffer;
 112     UINT8                   *LengthField;
 113     UINT32                  Length;
 114     UINT32                  TotalLength;
 115     UINT32                  SizeOfLengthField;
 116     UINT16                  Depth;
 117     UINT8                   Flags;

 119 } DT_SUBTABLE;


 122 /*
 123  * Globals
 124  */
```

```
126 /* List of all field names and values from the input source */

128 DT_EXTERN DT_FIELD          DT_INIT_GLOBAL (*Gbl_FieldList, NULL);

130 /* List of all compiled tables and subtables */

132 DT_EXTERN DT_SUBTABLE       DT_INIT_GLOBAL (*Gbl_RootTable, NULL);

134 /* Stack for subtables */

136 DT_EXTERN DT_SUBTABLE       DT_INIT_GLOBAL (*Gbl_SubtableStack, NULL);

138 /* List for defined labels */

140 DT_EXTERN DT_FIELD          DT_INIT_GLOBAL (*Gbl_LabelList, NULL);

142 /* Current offset within the binary output table */

144 DT_EXTERN UINT32            DT_INIT_GLOBAL (Gbl_CurrentTableOffset, 0);


147 /* dtcompiler - main module */

149 ACPI_STATUS
150 DtCompileTable (
151     DT_FIELD                **Field,
152     ACPI_DMTABLE_INFO       *Info,
153     DT_SUBTABLE             **RetSubtable,
154     BOOLEAN                 Required);


157 /* dtio - binary and text input/output */

159 UINT32
160 DtGetNextLine (
161     FILE                    *Handle);

163 DT_FIELD *
164 DtScanFile (
165     FILE                    *Handle);

167 void
168 DtOutputBinary (
169     DT_SUBTABLE             *RootTable);

171 void
172 DtDumpSubtableList (
173     void);

175 void
176 DtDumpFieldList (
177     DT_FIELD                *Field);

179 void
180 DtWriteFieldToListing (
181     UINT8                   *Buffer,
182     DT_FIELD                *Field,
183     UINT32                  Length);

185 void
186 DtWriteTableToListing (
187     void);


190 /* dtsubtable - compile subtables */
```



```
192 void
193 DtCreateSubtable (
194     UINT8                   *Buffer,
195     UINT32                  Length,
196     DT_SUBTABLE             **RetSubtable);

198 UINT32
199 DtGetSubtableLength (
200     DT_FIELD                *Field,
201     ACPI_DMTABLE_INFO       *Info);

203 void
204 DtSetSubtableLength (
205     DT_SUBTABLE             *Subtable);

207 void
208 DtPushSubtable (
209     DT_SUBTABLE             *Subtable);

211 void
212 DtPopSubtable (
213     void);

215 DT_SUBTABLE *
216 DtPeekSubtable (
217     void);

219 void
220 DtInsertSubtable (
221     DT_SUBTABLE             *ParentTable,
222     DT_SUBTABLE             *Subtable);

224 DT_SUBTABLE *
225 DtGetNextSubtable (
226     DT_SUBTABLE             *ParentTable,
227     DT_SUBTABLE             *ChildTable);

229 DT_SUBTABLE *
230 DtGetParentSubtable (
231     DT_SUBTABLE             *Subtable);


234 /* dtexpress - Integer expressions and labels */

236 ACPI_STATUS
237 DtResolveIntegerExpression (
238     DT_FIELD                *Field,
239     UINT64                  *ReturnValue);

241 UINT64
242 DtDoOperator (
243     UINT64                  LeftValue,
244     UINT32                  Operator,
245     UINT64                  RightValue);

247 UINT64
248 DtResolveLabel (
249     char                    *LabelString);

251 void
252 DtDetectAllLabels (
253     DT_FIELD                *FieldList);


256 /* dtfield - Compile individual fields within a table */
```

```
258 void
259 DtCompileOneField (
260     UINT8                   *Buffer,
261     DT_FIELD                *Field,
262     UINT32                  ByteLength,
263     UINT8                   Type,
264     UINT8                   Flags);

266 void
267 DtCompileInteger (
268     UINT8                   *Buffer,
269     DT_FIELD                *Field,
270     UINT32                  ByteLength,
271     UINT8                   Flags);

273 UINT32
274 DtCompileBuffer (
275     UINT8                   *Buffer,
276     char                    *Value,
277     DT_FIELD                *Field,
278     UINT32                  ByteLength);

280 void
281 DtCompileFlag (
282     UINT8                   *Buffer,
283     DT_FIELD                *Field,
284     ACPI_DMTABLE_INFO       *Info);


287 /* dtparser - lex/yacc files */

289 UINT64
290 DtEvaluateExpression (
291     char                    *ExprString);

293 int
294 DtInitLexer (
295     char                    *String);

297 void
298 DtTerminateLexer (
299     void);

301 char *
302 DtGetOpName (
303     UINT32                  ParseOpcode);


306 /* dtutils - Miscellaneous utilities */

308 typedef
309 void (*DT_WALK_CALLBACK) (
310     DT_SUBTABLE             *Subtable,
311     void                    *Context,
312     void                    *ReturnValue);

314 void
315 DtWalkTableTree (
316     DT_SUBTABLE             *StartTable,
317     DT_WALK_CALLBACK        UserFunction,
318     void                    *Context,
319     void                    *ReturnValue);

321 void
322 DtError (
323     UINT8                   Level,
```



```
324     UINT8                   MessageId,
325     DT_FIELD                *FieldObject,
326     char                    *ExtraMessage);

328 void
329 DtNameError (
330     UINT8                   Level,
331     UINT8                   MessageId,
332     DT_FIELD                *FieldObject,
333     char                    *ExtraMessage);

335 void
336 DtFatal (
337     UINT8                   MessageId,
338     DT_FIELD                *FieldObject,
339     char                    *ExtraMessage);

341 ACPI_STATUS
342 DtStrtoul64 (
343     char                    *String,
344     UINT64                  *ReturnInteger);

346 UINT32
347 DtGetFileSize (
348     FILE                    *Handle);

350 char*
351 DtGetFieldValue (
352     DT_FIELD                *Field);

354 UINT8
355 DtGetFieldType (
356     ACPI_DMTABLE_INFO       *Info);

358 UINT32
359 DtGetBufferLength (
360     char                    *Buffer);

362 UINT32
363 DtGetFieldLength (
364     DT_FIELD                *Field,
365     ACPI_DMTABLE_INFO       *Info);

367 void
368 DtSetTableChecksum (
369     UINT8                   *ChecksumPointer);

371 void
372 DtSetTableLength(
373     void);

375 void
376 DtFreeFieldList (
377     void);


380 /* dttable - individual table compilation */

382 ACPI_STATUS
383 DtCompileFacs (
384     DT_FIELD                **PFieldList);

386 ACPI_STATUS
387 DtCompileRsdp (
388     DT_FIELD                **PFieldList);
```

```
390 ACPI_STATUS
391 DtCompileAsf (
392     void                    **PFieldList);

394 ACPI_STATUS
395 DtCompileCpep (
396     void                    **PFieldList);

398 ACPI_STATUS
399 DtCompileCsrt (
400     void                    **PFieldList);

402 ACPI_STATUS
403 DtCompileDbg2 (
404     void                    **PFieldList);

406 ACPI_STATUS
407 DtCompileDmar (
408     void                    **PFieldList);

410 ACPI_STATUS
411 DtCompileEinj (
412     void                    **PFieldList);

414 ACPI_STATUS
415 DtCompileErst (
416     void                    **PFieldList);

418 ACPI_STATUS
419 DtCompileFadt (
420     void                    **PFieldList);

422 ACPI_STATUS
423 DtCompileFpdt (
424     void                    **PFieldList);

426 ACPI_STATUS
427 DtCompileHest (
428     void                    **PFieldList);

430 ACPI_STATUS
431 DtCompileIvrs (
432     void                    **PFieldList);

434 ACPI_STATUS
435 DtCompileMadt (
436     void                    **PFieldList);

438 ACPI_STATUS
439 DtCompileMcfg (
440     void                    **PFieldList);

442 ACPI_STATUS
443 DtCompileMpst (
444     void                    **PFieldList);

446 ACPI_STATUS
447 DtCompileMsct (
448     void                    **PFieldList);

450 ACPI_STATUS
451 DtCompileMtmr (
452     void                    **PFieldList);

454 ACPI_STATUS
455 DtCompilePmtt (
```



```
456     void                    **PFieldList);

458 ACPI_STATUS
459 DtCompilePcct (
460     void                    **PFieldList);

462 ACPI_STATUS
463 DtCompileRsdt (
464     void                    **PFieldList);

466 ACPI_STATUS
467 DtCompileS3pt (
468     DT_FIELD                **PFieldList);

470 ACPI_STATUS
471 DtCompileSlic (
472     void                    **PFieldList);

474 ACPI_STATUS
475 DtCompileSlit (
476     void                    **PFieldList);

478 ACPI_STATUS
479 DtCompileSrat (
480     void                    **PFieldList);

482 ACPI_STATUS
483 DtCompileUefi (
484     void                    **PFieldList);

486 ACPI_STATUS
487 DtCompileVrtc (
488     void                    **PFieldList);

490 ACPI_STATUS
491 DtCompileWdat (
492     void                    **PFieldList);

494 ACPI_STATUS
495 DtCompileXsdt (
496     void                    **PFieldList);

498 ACPI_STATUS
499 DtCompileGeneric (
500     void                    **PFieldList);

502 ACPI_DMTABLE_INFO *
503 DtGetGenericTableInfo (
504     char                    *Name);

506 /* ACPI Table templates */

508 extern const unsigned char  TemplateAsf[];
509 extern const unsigned char  TemplateBoot[];
510 extern const unsigned char  TemplateBert[];
511 extern const unsigned char  TemplateBgrt[];
512 extern const unsigned char  TemplateCpep[];
513 extern const unsigned char  TemplateCsrt[];
514 extern const unsigned char  TemplateDbg2[];
515 extern const unsigned char  TemplateDbgp[];
516 extern const unsigned char  TemplateDmar[];
517 extern const unsigned char  TemplateEcdt[];
518 extern const unsigned char  TemplateEinj[];
519 extern const unsigned char  TemplateErst[];
520 extern const unsigned char  TemplateFadt[];
521 extern const unsigned char  TemplateFpdt[];
```

```
522 extern const unsigned char  TemplateGtdt[];
523 extern const unsigned char  TemplateHest[];
524 extern const unsigned char  TemplateHpet[];
525 extern const unsigned char  TemplateIvrs[];
526 extern const unsigned char  TemplateMadt[];
527 extern const unsigned char  TemplateMcfg[];
528 extern const unsigned char  TemplateMchi[];
529 extern const unsigned char  TemplateMpst[];
530 extern const unsigned char  TemplateMsct[];
531 extern const unsigned char  TemplateMtmr[];
532 extern const unsigned char  TemplatePcct[];
533 extern const unsigned char  TemplatePmtt[];
534 extern const unsigned char  TemplateRsdt[];
535 extern const unsigned char  TemplateS3pt[];
536 extern const unsigned char  TemplateSbst[];
537 extern const unsigned char  TemplateSlic[];
538 extern const unsigned char  TemplateSlit[];
539 extern const unsigned char  TemplateSpcr[];
540 extern const unsigned char  TemplateSpmi[];
541 extern const unsigned char  TemplateSrat[];
542 extern const unsigned char  TemplateTcpa[];
543 extern const unsigned char  TemplateTpm2[];
544 extern const unsigned char  TemplateUefi[];
545 extern const unsigned char  TemplateVrtc[];
546 extern const unsigned char  TemplateWaet[];
547 extern const unsigned char  TemplateWdat[];
548 extern const unsigned char  TemplateWddt[];
549 extern const unsigned char  TemplateWdrt[];
550 extern const unsigned char  TemplateXsdt[];

552 #endif
```

```
**********************************************************
   10905 Thu Dec 26 13:48:37 2013
new/usr/src/common/acpica/compiler/dtexpress.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dtexpress.c - Support for integer expressions and labels
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTEXPRESS_C__

#include "aslcompiler.h"
#include "dtcompiler.h"
#include "dtparser.y.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtexpress")


/* Local prototypes */

static void
DtInsertLabelField (
    DT_FIELD                *Field);

static DT_FIELD *
```



```
DtLookupLabel (
    char                    *Name);

/* Global used for errors during parse and related functions */

DT_FIELD                *Gbl_CurrentField;


/******************************************************************************
 *
 * FUNCTION:    DtResolveIntegerExpression
 *
 * PARAMETERS:  Field               - Field object with Integer expression
 *              ReturnValue         - Where the integer is returned
 *
 * RETURN:      Status, and the resolved 64-bit integer value
 *
 * DESCRIPTION: Resolve an integer expression to a single value. Supports
 *              both integer constants and labels.
 *
 *****************************************************************************/

ACPI_STATUS
DtResolveIntegerExpression (
    DT_FIELD                *Field,
    UINT64                  *ReturnValue)
{
    UINT64                  Result;


    DbgPrint (ASL_DEBUG_OUTPUT, "Full Integer expression: %s\n",
        Field->Value);

    Gbl_CurrentField = Field;

    Result = DtEvaluateExpression (Field->Value);
    *ReturnValue = Result;
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtDoOperator
 *
 * PARAMETERS:  LeftValue           - First 64-bit operand
 *              Operator            - Parse token for the operator (EXPOP_*)
 *              RightValue          - Second 64-bit operand
 *
 * RETURN:      64-bit result of the requested operation
 *
 * DESCRIPTION: Perform the various 64-bit integer math functions
 *
 *****************************************************************************/

UINT64
DtDoOperator (
    UINT64                  LeftValue,
    UINT32                  Operator,
    UINT64                  RightValue)
{
    UINT64                  Result;


    /* Perform the requested operation */
```

```
127     switch (Operator)
128     {
129     case EXPOP_ONES_COMPLIMENT:

131         Result = ~RightValue;
132         break;

134     case EXPOP_LOGICAL_NOT:

136         Result = !RightValue;
137         break;

139     case EXPOP_MULTIPLY:

141         Result = LeftValue * RightValue;
142         break;

144     case EXPOP_DIVIDE:

146         if (!RightValue)
147         {
148             DtError (ASL_ERROR, ASL_MSG_DIVIDE_BY_ZERO,
149                 Gbl_CurrentField, NULL);
150             return (0);
151         }
152         Result = LeftValue / RightValue;
153         break;

155     case EXPOP_MODULO:

157         if (!RightValue)
158         {
159             DtError (ASL_ERROR, ASL_MSG_DIVIDE_BY_ZERO,
160                 Gbl_CurrentField, NULL);
161             return (0);
162         }
163         Result = LeftValue % RightValue;
164         break;

166     case EXPOP_ADD:
167         Result = LeftValue + RightValue;
168         break;

170     case EXPOP_SUBTRACT:

172         Result = LeftValue - RightValue;
173         break;

175     case EXPOP_SHIFT_RIGHT:

177         Result = LeftValue >> RightValue;
178         break;

180     case EXPOP_SHIFT_LEFT:

182         Result = LeftValue << RightValue;
183         break;

185     case EXPOP_LESS:

187         Result = LeftValue < RightValue;
188         break;

190     case EXPOP_GREATER:

192         Result = LeftValue > RightValue;
```



```
193         break;

195     case EXPOP_LESS_EQUAL:

197         Result = LeftValue <= RightValue;
198         break;

200     case EXPOP_GREATER_EQUAL:

202         Result = LeftValue >= RightValue;
203         break;

205     case EXPOP_EQUAL:

207         Result = LeftValue == RightValue;
208         break;

210     case EXPOP_NOT_EQUAL:

212         Result = LeftValue != RightValue;
213         break;

215     case EXPOP_AND:

217         Result = LeftValue & RightValue;
218         break;

220     case EXPOP_XOR:

222         Result = LeftValue ^ RightValue;
223         break;

225     case EXPOP_OR:

227         Result = LeftValue | RightValue;
228         break;

230     case EXPOP_LOGICAL_AND:

232         Result = LeftValue && RightValue;
233         break;

235     case EXPOP_LOGICAL_OR:

237         Result = LeftValue || RightValue;
238         break;

240    default:

242         /* Unknown operator */

244         DtFatal (ASL_MSG_INVALID_EXPRESSION,
245             Gbl_CurrentField, NULL);
246         return (0);
247     }

249     DbgPrint (ASL_DEBUG_OUTPUT,
250         "IntegerEval: (%8.8X%8.8X %s %8.8X%8.8X) = %8.8X%8.8X\n",
251         ACPI_FORMAT_UINT64 (LeftValue),
252         DtGetOpName (Operator),
253         ACPI_FORMAT_UINT64 (RightValue),
254         ACPI_FORMAT_UINT64 (Result));

256     return (Result);
257 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    DtResolveLabel
 *
 * PARAMETERS:  LabelString         - Contains the label
 *
 * RETURN:      Table offset associated with the label
 *
 * DESCRIPTION: Lookup a lable and return its value.
 *
 ******************************************************************************/

UINT64
DtResolveLabel (
    char                    *LabelString)
{
    DT_FIELD                *LabelField;


    DbgPrint (ASL_DEBUG_OUTPUT, "Resolve Label: %s\n", LabelString);

    /* Resolve a label reference to an integer (table offset) */

    if (*LabelString != '$')
    {
        return (0);
    }

    LabelField = DtLookupLabel (LabelString);
    if (!LabelField)
    {
        DtError (ASL_ERROR, ASL_MSG_UNKNOWN_LABEL,
            Gbl_CurrentField, LabelString);
        return (0);
    }

    /* All we need from the label is the offset in the table */

    DbgPrint (ASL_DEBUG_OUTPUT, "Resolved Label: 0x%8.8X\n",
        LabelField->TableOffset);

    return (LabelField->TableOffset);
}


/*******************************************************************************
 *
 * FUNCTION:    DtDetectAllLabels
 *
 * PARAMETERS:  FieldList           - Field object at start of generic list
 *
 * RETURN:      None
 *
 * DESCRIPTION: Detect all labels in a list of "generic" opcodes (such as
 *              a UEFI table.) and insert them into the global label list.
 *
 ******************************************************************************/

void
DtDetectAllLabels (
    DT_FIELD                *FieldList)
{
    ACPI_DMTABLE_INFO       *Info;
    DT_FIELD                *GenericField;
    UINT32                  TableOffset;
```



```
    TableOffset = Gbl_CurrentTableOffset;
    GenericField = FieldList;

    /*
     * Process all "Label:" fields within the parse tree. We need
     * to know the offsets for all labels before we can compile
     * the parse tree in order to handle forward references. Traverse
     * tree and get/set all field lengths of all operators in order to
     * determine the label offsets.
     */
    while (GenericField)
    {
        Info = DtGetGenericTableInfo (GenericField->Name);
        if (Info)
        {
            /* Maintain table offsets */

            GenericField->TableOffset = TableOffset;
            TableOffset += DtGetFieldLength (GenericField, Info);

            /* Insert all labels in the global label list */

            if (Info->Opcode == ACPI_DMT_LABEL)
            {
                DtInsertLabelField (GenericField);
            }
        }

        GenericField = GenericField->Next;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    DtInsertLabelField
 *
 * PARAMETERS:  Field               - Field object with Label to be inserted
 *
 * RETURN:      None
 *
 * DESCRIPTION: Insert a label field into the global label list
 *
 ******************************************************************************/

static void
DtInsertLabelField (
    DT_FIELD                *Field)
{

    DbgPrint (ASL_DEBUG_OUTPUT,
        "DtInsertLabelField: Found Label : %s at output table offset %X\n",
        Field->Value, Field->TableOffset);

    Field->NextLabel = Gbl_LabelList;
    Gbl_LabelList = Field;
}


/*******************************************************************************
 *
 * FUNCTION:    DtLookupLabel
 *
 * PARAMETERS:  Name                - Label to be resolved
```

```
 391  *
 392  * RETURN:      Field object associated with the label
 393  *
 394  * DESCRIPTION: Lookup a label in the global label list. Used during the
 395  *              resolution of integer expressions.
 396  *
 397  *****************************************************************************/

 399 static DT_FIELD *
 400 DtLookupLabel (
 401     char                    *Name)
 402 {
 403     DT_FIELD                *LabelField;


 406     /* Skip a leading $ */

 408     if (*Name == '$')
 409     {
 410         Name++;
 411     }

 413     /* Search global list */

 415     LabelField = Gbl_LabelList;
 416     while (LabelField)
 417     {
 418         if (!ACPI_STRCMP (Name, LabelField->Value))
 419         {
 420             return (LabelField);
 421         }
 422         LabelField = LabelField->NextLabel;
 423     }

 425     return (NULL);
 426 }
```

```
**********************************************************
   15206 Thu Dec 26 13:48:37 2013
new/usr/src/common/acpica/compiler/dtfield.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dtfield.c - Code generation for individual source fields
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTFIELD_C__

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtfield")


/* Local prototypes */

static void
DtCompileString (
    UINT8                   *Buffer,
    DT_FIELD                *Field,
    UINT32                  ByteLength);
```



```
static void
DtCompileUnicode (
    UINT8                   *Buffer,
    DT_FIELD                *Field,
    UINT32                  ByteLength);

static ACPI_STATUS
DtCompileUuid (
    UINT8                   *Buffer,
    DT_FIELD                *Field,
    UINT32                  ByteLength);

static char *
DtNormalizeBuffer (
    char                    *Buffer,
    UINT32                  *Count);


/******************************************************************************
 *
 * FUNCTION:    DtCompileOneField
 *
 * PARAMETERS:  Buffer              - Output buffer
 *              Field               - Field to be compiled
 *              ByteLength          - Byte length of the field
 *              Type                - Field type
 *
 * RETURN:      None
 *
 * DESCRIPTION: Compile a field value to binary
 *
 *****************************************************************************/

void
DtCompileOneField (
    UINT8                   *Buffer,
    DT_FIELD                *Field,
    UINT32                  ByteLength,
    UINT8                   Type,
    UINT8                   Flags)
{
    ACPI_STATUS             Status;

    switch (Type)
    {
    case DT_FIELD_TYPE_INTEGER:

        DtCompileInteger (Buffer, Field, ByteLength, Flags);
        break;

    case DT_FIELD_TYPE_STRING:

        DtCompileString (Buffer, Field, ByteLength);
        break;

    case DT_FIELD_TYPE_UUID:

        Status = DtCompileUuid (Buffer, Field, ByteLength);
        if (ACPI_SUCCESS (Status))
        {
            break;
        }

        /* Fall through. */
```

```
126     case DT_FIELD_TYPE_BUFFER:

128         DtCompileBuffer (Buffer, Field->Value, Field, ByteLength);
129         break;

131     case DT_FIELD_TYPE_UNICODE:

133         DtCompileUnicode (Buffer, Field, ByteLength);
134         break;

136     case DT_FIELD_TYPE_DEVICE_PATH:

138         break;

140     default:

142         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field, "Invalid field type");
143         break;
144     }
145 }


148 /******************************************************************************
149  *
150  * FUNCTION:    DtCompileString
151  *
152  * PARAMETERS:  Buffer              - Output buffer
153  *              Field               - String to be copied to buffer
154  *              ByteLength          - Maximum length of string
155  *
156  * RETURN:      None
157  *
158  * DESCRIPTION: Copy string to the buffer
159  *
160  *****************************************************************************/

162 static void
163 DtCompileString (
164     UINT8                   *Buffer,
165     DT_FIELD                *Field,
166     UINT32                  ByteLength)
167 {
168     UINT32                  Length;


171     Length = ACPI_STRLEN (Field->Value);

173     /* Check if the string is too long for the field */

175     if (Length > ByteLength)
176     {
177         sprintf (MsgBuffer, "Maximum %u characters", ByteLength);
178         DtError (ASL_ERROR, ASL_MSG_STRING_LENGTH, Field, MsgBuffer);
179         Length = ByteLength;
180     }

182     ACPI_MEMCPY (Buffer, Field->Value, Length);
183 }


186 /******************************************************************************
187  *
188  * FUNCTION:    DtCompileUnicode
189  *
190  * PARAMETERS:  Buffer              - Output buffer
191  *              Field               - String to be copied to buffer
```



```
192  *              ByteLength          - Maximum length of string
193  *
194  * RETURN:      None
195  *
196  * DESCRIPTION: Convert ASCII string to Unicode string
197  *
198  * Note:  The Unicode string is 16 bits per character, no leading signature,
199  *        with a 16-bit terminating NULL.
200  *
201  *****************************************************************************/

203 static void
204 DtCompileUnicode (
205     UINT8                   *Buffer,
206     DT_FIELD                *Field,
207     UINT32                  ByteLength)
208 {
209     UINT32                  Count;
210     UINT32                  i;
211     char                    *AsciiString;
212     UINT16                  *UnicodeString;


215     AsciiString = Field->Value;
216     UnicodeString = (UINT16 *) Buffer;
217     Count = ACPI_STRLEN (AsciiString) + 1;

219     /* Convert to Unicode string (including null terminator) */

221     for (i = 0; i < Count; i++)
222     {
223         UnicodeString[i] = (UINT16) AsciiString[i];
224     }
225 }


228 /******************************************************************************
229  *
230  * FUNCTION:    DtCompileUuid
231  *
232  * PARAMETERS:  Buffer              - Output buffer
233  *              Field               - String to be copied to buffer
234  *              ByteLength          - Maximum length of string
235  *
236  * RETURN:      None
237  *
238  * DESCRIPTION: Convert UUID string to 16-byte buffer
239  *
240  *****************************************************************************/

242 static ACPI_STATUS
243 DtCompileUuid (
244     UINT8                   *Buffer,
245     DT_FIELD                *Field,
246     UINT32                  ByteLength)
247 {
248     char                    *InString;
249     ACPI_STATUS             Status;


252     InString = Field->Value;

254     Status = AuValidateUuid (InString);
255     if (ACPI_FAILURE (Status))
256     {
257         sprintf (MsgBuffer, "%s", Field->Value);
```

```
258         DtNameError (ASL_ERROR, ASL_MSG_INVALID_UUID, Field, MsgBuffer);
259     }
260     else
261     {
262         Status = AuConvertStringToUuid (InString, (char *) Buffer);
263     }

265     return (Status);
266 }


269 /******************************************************************************
270  *
271  * FUNCTION:    DtCompileInteger
272  *
273  * PARAMETERS:  Buffer              - Output buffer
274  *              Field               - Field obj with Integer to be compiled
275  *              ByteLength          - Byte length of the integer
276  *              Flags               - Additional compile info
277  *
278  * RETURN:      None
279  *
280  * DESCRIPTION: Compile an integer. Supports integer expressions with C-style
281  *              operators.
282  *
283  *****************************************************************************/

285 void
286 DtCompileInteger (
287     UINT8                   *Buffer,
288     DT_FIELD                *Field,
289     UINT32                  ByteLength,
290     UINT8                   Flags)
291 {
292     UINT64                  Value;
293     UINT64                  MaxValue;
294     ACPI_STATUS             Status;


297     /* Output buffer byte length must be in range 1-8 */

299     if ((ByteLength > 8) || (ByteLength == 0))
300     {
301         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field,
302             "Invalid internal Byte length");
303         return;
304     }

306     /* Resolve integer expression to a single integer value */

308     Status = DtResolveIntegerExpression (Field, &Value);
309     if (ACPI_FAILURE (Status))
310     {
311         return;
312     }

314     /* Ensure that reserved fields are set to zero */
315     /* TBD: should we set to zero, or just make this an ERROR? */
316     /* TBD: Probably better to use a flag */

318     if (!ACPI_STRCMP (Field->Name, "Reserved") &&
319         (Value != 0))
320     {
321         DtError (ASL_WARNING, ASL_MSG_RESERVED_VALUE, Field,
322             "Setting to zero");
323         Value = 0;
```



```
324     }

326     /* Check if the value must be non-zero */

328     if ((Value == 0) && (Flags & DT_NON_ZERO))
329     {
330         DtError (ASL_ERROR, ASL_MSG_ZERO_VALUE, Field, NULL);
331     }

333     /*
334      * Generate the maximum value for the data type (ByteLength)
335      * Note: construct chosen for maximum portability
336      */
337     MaxValue = ((UINT64) (-1)) >> (64 - (ByteLength * 8));

339     /* Validate that the input value is within range of the target */

341     if (Value > MaxValue)
342     {
343         sprintf (MsgBuffer, "%8.8X%8.8X - max %u bytes",
344             ACPI_FORMAT_UINT64 (Value), ByteLength);
345         DtError (ASL_ERROR, ASL_MSG_INTEGER_SIZE, Field, MsgBuffer);
346     }

348     ACPI_MEMCPY (Buffer, &Value, ByteLength);
349     return;
350 }


353 /******************************************************************************
354  *
355  * FUNCTION:    DtNormalizeBuffer
356  *
357  * PARAMETERS:  Buffer              - Input buffer
358  *              Count               - Output the count of hex number in
359  *                                    the Buffer
360  *
361  * RETURN:      The normalized buffer, freed by caller
362  *
363  * DESCRIPTION: [1A,2B,3C,4D] or 1A, 2B, 3C, 4D will be normalized
364  *              to 1A 2B 3C 4D
365  *
366  *****************************************************************************/

368 static char *
369 DtNormalizeBuffer (
370     char                    *Buffer,
371     UINT32                  *Count)
372 {
373     char                    *NewBuffer;
374     char                    *TmpBuffer;
375     UINT32                  BufferCount = 0;
376     BOOLEAN                 Separator = TRUE;
377     char                    c;


380     NewBuffer = UtLocalCalloc (ACPI_STRLEN (Buffer) + 1);
381     TmpBuffer = NewBuffer;

383     while ((c = *Buffer++))
384     {
385         switch (c)
386         {
387         /* Valid separators */

389         case '[':
```

```
390         case ']':
391         case ' ':
392         case ',':

394             Separator = TRUE;
395             break;

397         default:

399             if (Separator)
400             {
401                 /* Insert blank as the standard separator */

403                 if (NewBuffer[0])
404                 {
405                     *TmpBuffer++ = ' ';
406                     BufferCount++;
407                 }

409                 Separator = FALSE;
410             }

412             *TmpBuffer++ = c;
413             break;
414         }
415     }

417     *Count = BufferCount + 1;
418     return (NewBuffer);
419 }


422 /******************************************************************************
423  *
424  * FUNCTION:    DtCompileBuffer
425  *
426  * PARAMETERS:  Buffer              - Output buffer
427  *              StringValue         - Integer list to be compiled
428  *              Field               - Current field object
429  *              ByteLength          - Byte length of the integer list
430  *
431  * RETURN:      Count of remaining data in the input list
432  *
433  * DESCRIPTION: Compile and pack an integer list, for example
434  *              "AA 1F 20 3B" ==> Buffer[] = {0xAA,0x1F,0x20,0x3B}
435  *
436  *****************************************************************************/

438 UINT32
439 DtCompileBuffer (
440     UINT8                   *Buffer,
441     char                    *StringValue,
442     DT_FIELD                *Field,
443     UINT32                  ByteLength)
444 {
445     ACPI_STATUS             Status;
446     char                    Hex[3];
447     UINT64                  Value;
448     UINT32                  i;
449     UINT32                  Count;


452     /* Allow several different types of value separators */

454     StringValue = DtNormalizeBuffer (StringValue, &Count);
```



```
456     Hex[2] = 0;
457     for (i = 0; i < Count; i++)
458     {
459         /* Each element of StringValue is three chars */

461         Hex[0] = StringValue[(3 * i)];
462         Hex[1] = StringValue[(3 * i) + 1];

464         /* Convert one hex byte */

466         Value = 0;
467         Status = DtStrtoul64 (Hex, &Value);
468         if (ACPI_FAILURE (Status))
469         {
470             DtError (ASL_ERROR, ASL_MSG_BUFFER_ELEMENT, Field, MsgBuffer);
471             goto Exit;
472         }

474         Buffer[i] = (UINT8) Value;
475     }

477 Exit:
478     ACPI_FREE (StringValue);
479     return (ByteLength - Count);
480 }


483 /******************************************************************************
484  *
485  * FUNCTION:    DtCompileFlag
486  *
487  * PARAMETERS:  Buffer              - Output buffer
488  *              Field               - Field to be compiled
489  *              Info                - Flag info
490  *
491  * RETURN:
492  *
493  * DESCRIPTION: Compile a flag
494  *
495  *****************************************************************************/

497 void
498 DtCompileFlag (
499     UINT8                   *Buffer,
500     DT_FIELD                *Field,
501     ACPI_DMTABLE_INFO       *Info)
502 {
503     UINT64                  Value = 0;
504     UINT32                  BitLength = 1;
505     UINT8                   BitPosition = 0;
506     ACPI_STATUS             Status;


509     Status = DtStrtoul64 (Field->Value, &Value);
510     if (ACPI_FAILURE (Status))
511     {
512         DtError (ASL_ERROR, ASL_MSG_INVALID_HEX_INTEGER, Field, NULL);
513     }

515     switch (Info->Opcode)
516     {
517     case ACPI_DMT_FLAG0:
518     case ACPI_DMT_FLAG1:
519     case ACPI_DMT_FLAG2:
520     case ACPI_DMT_FLAG3:
521     case ACPI_DMT_FLAG4:
```

```
522     case ACPI_DMT_FLAG5:
523     case ACPI_DMT_FLAG6:
524     case ACPI_DMT_FLAG7:

526         BitPosition = Info->Opcode;
527         BitLength = 1;
528         break;

530     case ACPI_DMT_FLAGS0:

532         BitPosition = 0;
533         BitLength = 2;
534         break;


537     case ACPI_DMT_FLAGS1:

539         BitPosition = 1;
540         BitLength = 2;
541         break;


544     case ACPI_DMT_FLAGS2:

546         BitPosition = 2;
547         BitLength = 2;
548         break;

550     case ACPI_DMT_FLAGS4:

552         BitPosition = 4;
553         BitLength = 2;
554         break;

556     default:

558         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field, "Invalid flag opcode");
559         break;
560     }

562     /* Check range of the input flag value */

564     if (Value >= ((UINT64) 1 << BitLength))
565     {
566         sprintf (MsgBuffer, "Maximum %u bit", BitLength);
567         DtError (ASL_ERROR, ASL_MSG_FLAG_VALUE, Field, MsgBuffer);
568         Value = 0;
569     }

571     *Buffer |= (UINT8) (Value << BitPosition);
572 }
```

```
**********************************************************
   28481 Thu Dec 26 13:48:37 2013
new/usr/src/common/acpica/compiler/dtio.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dtio.c - File I/O support for data table compiler
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTIO_C__

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtio")


/* Local prototypes */

static char *
DtTrim (
    char                    *String);

static void
DtLinkField (
```



```
    DT_FIELD                *Field);

static ACPI_STATUS
DtParseLine (
    char                    *LineBuffer,
    UINT32                  Line,
    UINT32                  Offset);

static void
DtWriteBinary (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
    void                    *ReturnValue);

static void
DtDumpBuffer (
    UINT32                  FileId,
    UINT8                   *Buffer,
    UINT32                  Offset,
    UINT32                  Length);

static void
DtDumpSubtableInfo (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
    void                    *ReturnValue);

static void
DtDumpSubtableTree (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
    void                    *ReturnValue);


/* States for DtGetNextLine */

#define DT_NORMAL_TEXT              0
#define DT_START_QUOTED_STRING      1
#define DT_START_COMMENT            2
#define DT_SLASH_ASTERISK_COMMENT   3
#define DT_SLASH_SLASH_COMMENT      4
#define DT_END_COMMENT              5
#define DT_MERGE_LINES              6
#define DT_ESCAPE_SEQUENCE          7

static UINT32  Gbl_NextLineOffset;


/******************************************************************************
 *
 * FUNCTION:    DtTrim
 *
 * PARAMETERS:  String              - Current source code line to trim
 *
 * RETURN:      Trimmed line. Must be freed by caller.
 *
 * DESCRIPTION: Trim left and right spaces
 *
 *****************************************************************************/

static char *
DtTrim (
    char                    *String)
{
    char                    *Start;
    char                    *End;
```

```
127     char                    *ReturnString;
128     ACPI_SIZE               Length;


131     /* Skip lines that start with a space */

133     if (!ACPI_STRCMP (String, " "))
134     {
135         ReturnString = UtLocalCalloc (1);
136         return (ReturnString);
137     }

139     /* Setup pointers to start and end of input string */

141     Start = String;
142     End = String + ACPI_STRLEN (String) - 1;

144     /* Find first non-whitespace character */

146     while ((Start <= End) && ((*Start == ' ') || (*Start == '\t')))
147     {
148         Start++;
149     }

151     /* Find last non-space character */

153     while (End >= Start)
154     {
155         if (*End == '\r' || *End == '\n')
156         {
157             End--;
158             continue;
159         }

161         if (*End != ' ')
162         {
163             break;
164         }

166         End--;
167     }

169     /* Remove any quotes around the string */

171     if (*Start == '\"')
172     {
173         Start++;
174     }
175     if (*End == '\"')
176     {
177         End--;
178     }

180     /* Create the trimmed return string */

182     Length = ACPI_PTR_DIFF (End, Start) + 1;
183     ReturnString = UtLocalCalloc (Length + 1);
184     if (ACPI_STRLEN (Start))
185     {
186         ACPI_STRNCPY (ReturnString, Start, Length);
187     }

189     ReturnString[Length] = 0;
190     return (ReturnString);
191 }
```



```
194 /******************************************************************************
195  *
196  * FUNCTION:    DtLinkField
197  *
198  * PARAMETERS:  Field               - New field object to link
199  *
200  * RETURN:      None
201  *
202  * DESCRIPTION: Link one field name and value to the list
203  *
204  *****************************************************************************/

206 static void
207 DtLinkField (
208     DT_FIELD                *Field)
209 {
210     DT_FIELD                *Prev;
211     DT_FIELD                *Next;


214     Prev = Next = Gbl_FieldList;

216     while (Next)
217     {
218         Prev = Next;
219         Next = Next->Next;
220     }

222     if (Prev)
223     {
224         Prev->Next = Field;
225     }
226     else
227     {
228         Gbl_FieldList = Field;
229     }
230 }


233 /******************************************************************************
234  *
235  * FUNCTION:    DtParseLine
236  *
237  * PARAMETERS:  LineBuffer          - Current source code line
238  *              Line                - Current line number in the source
239  *              Offset              - Current byte offset of the line
240  *
241  * RETURN:      Status
242  *
243  * DESCRIPTION: Parse one source line
244  *
245  *****************************************************************************/

247 static ACPI_STATUS
248 DtParseLine (
249     char                    *LineBuffer,
250     UINT32                  Line,
251     UINT32                  Offset)
252 {
253     char                    *Start;
254     char                    *End;
255     char                    *TmpName;
256     char                    *TmpValue;
257     char                    *Name;
258     char                    *Value;
```

```
259     char                    *Colon;
260     UINT32                  Length;
261     DT_FIELD                *Field;
262     UINT32                  Column;
263     UINT32                  NameColumn;
264     BOOLEAN                 IsNullString = FALSE;


267     if (!LineBuffer)
268     {
269         return (AE_OK);
270     }

272     /* All lines after "Raw Table Data" are ingored */

274     if (strstr (LineBuffer, ACPI_RAW_TABLE_DATA_HEADER))
275     {
276         return (AE_NOT_FOUND);
277     }

279     Colon = strchr (LineBuffer, ':');
280     if (!Colon)
281     {
282         return (AE_OK);
283     }

285     Start = LineBuffer;
286     End = Colon;

288     while (Start < Colon)
289     {
290         if (*Start == '[')
291         {
292             /* Found left bracket, go to the right bracket */

294             while (Start < Colon && *Start != ']')
295             {
296                 Start++;
297             }
298         }
299         else if (*Start != ' ')
300         {
301             break;
302         }

304         Start++;
305     }

307     /*
308      * There are two column values. One for the field name,
309      * and one for the field value.
310      */
311     Column = ACPI_PTR_DIFF (Colon, LineBuffer) + 3;
312     NameColumn = ACPI_PTR_DIFF (Start, LineBuffer) + 1;

314     Length = ACPI_PTR_DIFF (End, Start);

316     TmpName = UtLocalCalloc (Length + 1);
317     ACPI_STRNCPY (TmpName, Start, Length);
318     Name = DtTrim (TmpName);
319     ACPI_FREE (TmpName);

321     Start = End = (Colon + 1);
322     while (*End)
323     {
324         /* Found left quotation, go to the right quotation and break */
```



```
326         if (*End == '"')
327         {
328             End++;

330             /* Check for an explicit null string */

332             if (*End == '"')
333             {
334                 IsNullString = TRUE;
335             }
336             while (*End && (*End != '"'))
337             {
338                 End++;
339             }

341             End++;
342             break;
343         }

345         /*
346          * Special "comment" fields at line end, ignore them.
347          * Note: normal slash-slash and slash-asterisk comments are
348          * stripped already by the DtGetNextLine parser.
349          *
350          * TBD: Perhaps DtGetNextLine should parse the following type
351          * of comments also.
352          */
353         if (*End == '[')
354         {
355             End--;
356             break;
357         }
358         End++;
359     }

361     Length = ACPI_PTR_DIFF (End, Start);
362     TmpValue = UtLocalCalloc (Length + 1);

364     ACPI_STRNCPY (TmpValue, Start, Length);
365     Value = DtTrim (TmpValue);
366     ACPI_FREE (TmpValue);

368     /* Create a new field object only if we have a valid value field */

370     if ((Value && *Value) || IsNullString)
371     {
372         Field = UtLocalCalloc (sizeof (DT_FIELD));
373         Field->Name = Name;
374         Field->Value = Value;
375         Field->Line = Line;
376         Field->ByteOffset = Offset;
377         Field->NameColumn = NameColumn;
378         Field->Column = Column;

380         DtLinkField (Field);
381     }
382     else /* Ignore this field, it has no valid data */
383     {
384         ACPI_FREE (Name);
385         ACPI_FREE (Value);
386     }

388     return (AE_OK);
389 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    DtGetNextLine
 *
 * PARAMETERS:  Handle              - Open file handle for the source file
 *
 * RETURN:      Filled line buffer and offset of start-of-line (ASL_EOF on EOF)
 *
 * DESCRIPTION: Get the next valid source line. Removes all comments.
 *              Ignores empty lines.
 *
 * Handles both slash-asterisk and slash-slash comments.
 * Also, quoted strings, but no escapes within.
 *
 * Line is returned in Gbl_CurrentLineBuffer.
 * Line number in original file is returned in Gbl_CurrentLineNumber.
 *
 ******************************************************************************/

UINT32
DtGetNextLine (
    FILE                    *Handle)
{
    BOOLEAN                 LineNotAllBlanks = FALSE;
    UINT32                  State = DT_NORMAL_TEXT;
    UINT32                  CurrentLineOffset;
    UINT32                  i;
    int                     c;


    for (i = 0; ;)
    {
        /*
         * If line is too long, expand the line buffers. Also increases
         * Gbl_LineBufferSize.
         */
        if (i >= Gbl_LineBufferSize)
        {
            UtExpandLineBuffers ();
        }

        c = getc (Handle);
        if (c == EOF)
        {
            switch (State)
            {
            case DT_START_QUOTED_STRING:
            case DT_SLASH_ASTERISK_COMMENT:

                AcpiOsPrintf ("**** EOF within comment/string %u\n", State);
                break;

            default:

                break;
            }

            /* Standalone EOF is OK */

            if (i == 0)
            {
                return (ASL_EOF);
            }

            /*
```



```
             * Received an EOF in the middle of a line. Terminate the
             * line with a newline. The next call to this function will
             * return a standalone EOF. Thus, the upper parsing software
             * never has to deal with an EOF within a valid line (or
             * the last line does not get tossed on the floor.)
             */
            c = '\n';
            State = DT_NORMAL_TEXT;
        }

        switch (State)
        {
        case DT_NORMAL_TEXT:

            /* Normal text, insert char into line buffer */

            Gbl_CurrentLineBuffer[i] = (char) c;
            switch (c)
            {
            case '/':

                State = DT_START_COMMENT;
                break;

            case '"':

                State = DT_START_QUOTED_STRING;
                LineNotAllBlanks = TRUE;
                i++;
                break;

            case '\\':
                /*
                 * The continuation char MUST be last char on this line.
                 * Otherwise, it will be assumed to be a valid ASL char.
                 */
                State = DT_MERGE_LINES;
                break;

            case '\n':

                CurrentLineOffset = Gbl_NextLineOffset;
                Gbl_NextLineOffset = (UINT32) ftell (Handle);
                Gbl_CurrentLineNumber++;

                /*
                 * Exit if line is complete. Ignore empty lines (only \n)
                 * or lines that contain nothing but blanks.
                 */
                if ((i != 0) && LineNotAllBlanks)
                {
                    if ((i + 1) >= Gbl_LineBufferSize)
                    {
                        UtExpandLineBuffers ();
                    }

                    Gbl_CurrentLineBuffer[i+1] = 0; /* Terminate string */
                    return (CurrentLineOffset);
                }

                /* Toss this line and start a new one */

                i = 0;
                LineNotAllBlanks = FALSE;
                break;
```

```
523             default:
525                 if (c != ' ')
526                 {
527                     LineNotAllBlanks = TRUE;
528                 }
530                 i++;
531                 break;
532             }
533             break;
535         case DT_START_QUOTED_STRING:
537             /* Insert raw chars until end of quoted string */
539             Gbl_CurrentLineBuffer[i] = (char) c;
540             i++;
542             switch (c)
543             {
544             case '"':
546                 State = DT_NORMAL_TEXT;
547                 break;
549             case '\\':
551                 State = DT_ESCAPE_SEQUENCE;
552                 break;
554             case '\n':
556                 AcpiOsPrintf ("ERROR at line %u: Unterminated quoted string\n",
557                     Gbl_CurrentLineNumber++);
558                 State = DT_NORMAL_TEXT;
559                 break;
561             default:    /* Get next character */
563                 break;
564             }
565             break;
567         case DT_ESCAPE_SEQUENCE:
569             /* Just copy the escaped character. TBD: sufficient for table compil
571             Gbl_CurrentLineBuffer[i] = (char) c;
572             i++;
573             State = DT_START_QUOTED_STRING;
574             break;
576         case DT_START_COMMENT:
578             /* Open comment if this character is an asterisk or slash */
580             switch (c)
581             {
582             case '*':
584                 State = DT_SLASH_ASTERISK_COMMENT;
585                 break;
587             case '/':
```



```
589                 State = DT_SLASH_SLASH_COMMENT;
590                 break;
592             default:    /* Not a comment */
594                 i++;    /* Save the preceding slash */
595                 if (i >= Gbl_LineBufferSize)
596                 {
597                     UtExpandLineBuffers ();
598                 }
600                 Gbl_CurrentLineBuffer[i] = (char) c;
601                 i++;
602                 State = DT_NORMAL_TEXT;
603                 break;
604             }
605             break;
607         case DT_SLASH_ASTERISK_COMMENT:
609             /* Ignore chars until an asterisk-slash is found */
611             switch (c)
612             {
613             case '\n':
615                 Gbl_NextLineOffset = (UINT32) ftell (Handle);
616                 Gbl_CurrentLineNumber++;
617                 break;
619             case '*':
621                 State = DT_END_COMMENT;
622                 break;
624             default:
626                 break;
627             }
628             break;
630         case DT_SLASH_SLASH_COMMENT:
632             /* Ignore chars until end-of-line */
634             if (c == '\n')
635             {
636                 /* We will exit via the NORMAL_TEXT path */
638                 ungetc (c, Handle);
639                 State = DT_NORMAL_TEXT;
640             }
641             break;
643         case DT_END_COMMENT:
645             /* End comment if this char is a slash */
647             switch (c)
648             {
649             case '/':
651                 State = DT_NORMAL_TEXT;
652                 break;
654             case '\n':
```

```
656                 CurrentLineOffset = Gbl_NextLineOffset;
657                 Gbl_NextLineOffset = (UINT32) ftell (Handle);
658                 Gbl_CurrentLineNumber++;
659                 break;

661             case '*':

663                 /* Consume all adjacent asterisks */
664                 break;

666             default:

668                 State = DT_SLASH_ASTERISK_COMMENT;
669                 break;
670             }
671             break;

673         case DT_MERGE_LINES:

675             if (c != '\n')
676             {
677                 /*
678                  * This is not a continuation backslash, it is a normal
679                  * normal ASL backslash - for example: Scope(\_SB_)
680                  */
681                 i++; /* Keep the backslash that is already in the buffer */

683                 ungetc (c, Handle);
684                 State = DT_NORMAL_TEXT;
685             }
686             else
687             {
688                 /*
689                  * This is a continuation line -- a backlash followed
690                  * immediately by a newline. Insert a space between the
691                  * lines (overwrite the backslash)
692                  */
693                 Gbl_CurrentLineBuffer[i] = ' ';
694                 i++;

696                 /* Ignore newline, this will merge the lines */

698                 CurrentLineOffset = Gbl_NextLineOffset;
699                 Gbl_NextLineOffset = (UINT32) ftell (Handle);
700                 Gbl_CurrentLineNumber++;
701                 State = DT_NORMAL_TEXT;
702             }
703             break;

705         default:

707             DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, "Unknown input state");
708             return (ASL_EOF);
709         }
710     }
711 }


714 /*******************************************************************************
715  *
716  * FUNCTION:    DtScanFile
717  *
718  * PARAMETERS:  Handle              - Open file handle for the source file
719  *
720  * RETURN:      Pointer to start of the constructed parse tree.
```



```
721  *
722  * DESCRIPTION: Scan source file, link all field names and values
723  *              to the global parse tree: Gbl_FieldList
724  *
725  ******************************************************************************/

727 DT_FIELD *
728 DtScanFile (
729     FILE                    *Handle)
730 {
731     ACPI_STATUS             Status;
732     UINT32                  Offset;


735     ACPI_FUNCTION_NAME (DtScanFile);


738     /* Get the file size */

740     Gbl_InputByteCount = DtGetFileSize (Handle);

742     Gbl_CurrentLineNumber = 0;
743     Gbl_CurrentLineOffset = 0;
744     Gbl_NextLineOffset = 0;

746     /* Scan line-by-line */

748     while ((Offset = DtGetNextLine (Handle)) != ASL_EOF)
749     {
750         ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "Line %2.2u/%4.4X - %s",
751             Gbl_CurrentLineNumber, Offset, Gbl_CurrentLineBuffer));

753         Status = DtParseLine (Gbl_CurrentLineBuffer, Gbl_CurrentLineNumber, Offs
754         if (Status == AE_NOT_FOUND)
755         {
756             break;
757         }
758     }

760     /* Dump the parse tree if debug enabled */

762     DtDumpFieldList (Gbl_FieldList);
763     return (Gbl_FieldList);
764 }


767 /*
768  * Output functions
769  */

771 /*******************************************************************************
772  *
773  * FUNCTION:    DtWriteBinary
774  *
775  * PARAMETERS:  DT_WALK_CALLBACK
776  *
777  * RETURN:      Status
778  *
779  * DESCRIPTION: Write one subtable of a binary ACPI table
780  *
781  ******************************************************************************/

783 static void
784 DtWriteBinary (
785     DT_SUBTABLE             *Subtable,
786     void                    *Context,
```

```
    void                    *ReturnValue)
{

    FlWriteFile (ASL_FILE_AML_OUTPUT, Subtable->Buffer, Subtable->Length);
}


/*******************************************************************************
 *
 * FUNCTION:    DtOutputBinary
 *
 * PARAMETERS:
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Write entire binary ACPI table (result of compilation)
 *
 ******************************************************************************/

void
DtOutputBinary (
    DT_SUBTABLE             *RootTable)
{

    if (!RootTable)
    {
        return;
    }

    /* Walk the entire parse tree, emitting the binary data */

    DtWalkTableTree (RootTable, DtWriteBinary, NULL, NULL);
    Gbl_TableLength = DtGetFileSize (Gbl_Files[ASL_FILE_AML_OUTPUT].Handle);
}


/*
 * Listing support
 */

/*******************************************************************************
 *
 * FUNCTION:    DtDumpBuffer
 *
 * PARAMETERS:  FileID              - Where to write buffer data
 *              Buffer              - Buffer to dump
 *              Offset              - Offset in current table
 *              Length              - Buffer Length
 *
 * RETURN:      None
 *
 * DESCRIPTION: Another copy of DumpBuffer routine (unfortunately).
 *
 * TBD: merge dump buffer routines
 *
 ******************************************************************************/

static void
DtDumpBuffer (
    UINT32                  FileId,
    UINT8                   *Buffer,
    UINT32                  Offset,
    UINT32                  Length)
{
    UINT32                  i;
    UINT32                  j;
```



```
    UINT8                   BufChar;


    FlPrintFile (FileId, "Output: [%3.3Xh %4.4d %3d] ",
        Offset, Offset, Length);

    i = 0;
    while (i < Length)
    {
        if (i >= 16)
        {
            FlPrintFile (FileId, "%24s", "");
        }

        /* Print 16 hex chars */

        for (j = 0; j < 16;)
        {
            if (i + j >= Length)
            {
                /* Dump fill spaces */

                FlPrintFile (FileId, "   ");
                j++;
                continue;
            }

            FlPrintFile (FileId, "%02X ", Buffer[i+j]);
            j++;
        }

        FlPrintFile (FileId, " ");
        for (j = 0; j < 16; j++)
        {
            if (i + j >= Length)
            {
                FlPrintFile (FileId, "\n\n");
                return;
            }

            BufChar = Buffer[(ACPI_SIZE) i + j];
            if (ACPI_IS_PRINT (BufChar))
            {
                FlPrintFile (FileId, "%c", BufChar);
            }
            else
            {
                FlPrintFile (FileId, ".");
            }
        }

        /* Done with that line. */

        FlPrintFile (FileId, "\n");
        i += 16;
    }

    FlPrintFile (FileId, "\n\n");
}


/*******************************************************************************
 *
 * FUNCTION:    DtDumpFieldList
 *
 * PARAMETERS:  Field               - Root field
```

```
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the entire field list
 *
 ******************************************************************************/

void
DtDumpFieldList (
    DT_FIELD                *Field)
{

    if (!Gbl_DebugFlag || !Field)
    {
        return;
    }

    DbgPrint (ASL_DEBUG_OUTPUT,  "\nField List:\n"
        "LineNo   ByteOff  NameCol  Column   TableOff "
        "Flags    %32s : %s\n\n", "Name", "Value");
    while (Field)
    {
        DbgPrint (ASL_DEBUG_OUTPUT,
            "%.08X %.08X %.08X %.08X %.08X %.08X %32s : %s\n",
            Field->Line, Field->ByteOffset, Field->NameColumn,
            Field->Column, Field->TableOffset, Field->Flags,
            Field->Name, Field->Value);

        Field = Field->Next;
    }

    DbgPrint (ASL_DEBUG_OUTPUT,  "\n\n");
}


/*******************************************************************************
 *
 * FUNCTION:    DtDumpSubtableInfo, DtDumpSubtableTree
 *
 * PARAMETERS:  DT_WALK_CALLBACK
 *
 * RETURN:      None
 *
 * DESCRIPTION: Info - dump a subtable tree entry with extra information.
 *              Tree - dump a subtable tree formatted by depth indentation.
 *
 ******************************************************************************/

static void
DtDumpSubtableInfo (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
    void                    *ReturnValue)
{

    DbgPrint (ASL_DEBUG_OUTPUT,
        "[%.04X] %.08X %.08X %.08X %.08X %.08X %p %p %p\n",
        Subtable->Depth, Subtable->Length, Subtable->TotalLength,
        Subtable->SizeOfLengthField, Subtable->Flags, Subtable,
        Subtable->Parent, Subtable->Child, Subtable->Peer);
}

static void
DtDumpSubtableTree (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
```



```
    void                    *ReturnValue)
{

    DbgPrint (ASL_DEBUG_OUTPUT,
        "[%.04X] %*s%08X (%.02X) - (%.02X)\n",
        Subtable->Depth, (4 * Subtable->Depth), " ",
        Subtable, Subtable->Length, Subtable->TotalLength);
}


/*******************************************************************************
 *
 * FUNCTION:    DtDumpSubtableList
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the raw list of subtables with information, and also
 *              dump the subtable list in formatted tree format. Assists with
 *              the development of new table code.
 *
 ******************************************************************************/

void
DtDumpSubtableList (
    void)
{

    if (!Gbl_DebugFlag || !Gbl_RootTable)
    {
        return;
    }

    DbgPrint (ASL_DEBUG_OUTPUT,
        "Subtable Info:\n"
        "Depth  Length   TotalLen LenSize  Flags    "
        "This     Parent   Child    Peer\n\n");
    DtWalkTableTree (Gbl_RootTable, DtDumpSubtableInfo, NULL, NULL);

    DbgPrint (ASL_DEBUG_OUTPUT,
        "\nSubtable Tree: (Depth, Subtable, Length, TotalLength)\n\n");
    DtWalkTableTree (Gbl_RootTable, DtDumpSubtableTree, NULL, NULL);
}


/*******************************************************************************
 *
 * FUNCTION:    DtWriteFieldToListing
 *
 * PARAMETERS:  Buffer              - Contains the compiled data
 *              Field               - Field node for the input line
 *              Length              - Length of the output data
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write one field to the listing file (if listing is enabled).
 *
 ******************************************************************************/

void
DtWriteFieldToListing (
    UINT8                   *Buffer,
    DT_FIELD                *Field,
    UINT32                  Length)
{
```

```
1051     UINT8                  FileByte;


1054     if (!Gbl_ListingFlag || !Field)
1055     {
1056         return;
1057     }

1059     /* Dump the original source line */

1061     FlPrintFile (ASL_FILE_LISTING_OUTPUT, "Input:  ");
1062     FlSeekFile (ASL_FILE_INPUT, Field->ByteOffset);

1064     while (FlReadFile (ASL_FILE_INPUT, &FileByte, 1) == AE_OK)
1065     {
1066         FlWriteFile (ASL_FILE_LISTING_OUTPUT, &FileByte, 1);
1067         if (FileByte == '\n')
1068         {
1069             break;
1070         }
1071     }

1073     /* Dump the line as parsed and represented internally */

1075     FlPrintFile (ASL_FILE_LISTING_OUTPUT, "Parsed: %*s : %.64s",
1076         Field->Column-4, Field->Name, Field->Value);

1078     if (strlen (Field->Value) > 64)
1079     {
1080         FlPrintFile (ASL_FILE_LISTING_OUTPUT, "...Additional data, length 0x%X\n
1081             strlen (Field->Value));
1082     }
1083     FlPrintFile (ASL_FILE_LISTING_OUTPUT, "\n");

1085     /* Dump the hex data that will be output for this field */

1087     DtDumpBuffer (ASL_FILE_LISTING_OUTPUT, Buffer, Field->TableOffset, Length);
1088 }


1091 /*******************************************************************************
1092  *
1093  * FUNCTION:    DtWriteTableToListing
1094  *
1095  * PARAMETERS:  None
1096  *
1097  * RETURN:      None
1098  *
1099  * DESCRIPTION: Write the entire compiled table to the listing file
1100  *              in hex format
1101  *
1102  ******************************************************************************/

1104 void
1105 DtWriteTableToListing (
1106     void)
1107 {
1108     UINT8                  *Buffer;


1111     if (!Gbl_ListingFlag)
1112     {
1113         return;
1114     }

1116     /* Read the entire table from the output file */
```



```
1118     Buffer = UtLocalCalloc (Gbl_TableLength);
1119     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
1120     FlReadFile (ASL_FILE_AML_OUTPUT, Buffer, Gbl_TableLength);

1122     /* Dump the raw table data */

1124     AcpiOsRedirectOutput (Gbl_Files[ASL_FILE_LISTING_OUTPUT].Handle);

1126     AcpiOsPrintf ("\n%s: Length %d (0x%X)\n\n",
1127         ACPI_RAW_TABLE_DATA_HEADER, Gbl_TableLength, Gbl_TableLength);
1128     AcpiUtDumpBuffer (Buffer, Gbl_TableLength, DB_BYTE_DISPLAY, 0);

1130     AcpiOsRedirectOutput (stdout);
1131     ACPI_FREE (Buffer);
1132 }
```

```
**********************************************************
    4461 Thu Dec 26 13:48:37 2013
new/usr/src/common/acpica/compiler/dtparser.l
acpica-unix2-20130823
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: dtparser.l - Flex input file for table compiler lexer
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtparser.y.h"

#define YY_NO_INPUT     /* No file input, we use strings only */

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("dtscanner")
%}

%option noyywrap
%option nounput

Number          [0-9a-fA-F]+
HexNumber       0[xX][0-9a-fA-F]+
DecimalNumber   0[dD][0-9]+
LabelRef        $[a-zA-Z][0-9a-zA-Z]*
WhiteSpace      [ \t\v\r]+
```



```
NewLine         [\n]

%%

\(              return (EXPOP_PAREN_OPEN);
\)              return (EXPOP_PAREN_CLOSE);
\~              return (EXPOP_ONES_COMPLIMENT);
\!              return (EXPOP_LOGICAL_NOT);
\*              return (EXPOP_MULTIPLY);
\/              return (EXPOP_DIVIDE);
\%              return (EXPOP_MODULO);
\+              return (EXPOP_ADD);
\-              return (EXPOP_SUBTRACT);
">>"            return (EXPOP_SHIFT_RIGHT);
"<<"            return (EXPOP_SHIFT_LEFT);
\<              return (EXPOP_LESS);
\>              return (EXPOP_GREATER);
"<="            return (EXPOP_LESS_EQUAL);
">="            return (EXPOP_GREATER_EQUAL);
"=="            return (EXPOP_EQUAL);
"!="            return (EXPOP_NOT_EQUAL);
\&              return (EXPOP_AND);
\^              return (EXPOP_XOR);
\|              return (EXPOP_OR);
"&&"            return (EXPOP_LOGICAL_AND);
"||"            return (EXPOP_LOGICAL_OR);
<<EOF>>         return (EXPOP_EOF); /* null end-of-string */

{LabelRef}      return (EXPOP_LABEL);
{Number}        return (EXPOP_NUMBER);
{HexNumber}     return (EXPOP_HEX_NUMBER);
{NewLine}       return (EXPOP_NEW_LINE);
{WhiteSpace}    /* Ignore */

.               return (EXPOP_EOF);

%%

/*
 * Local support functions
 */
YY_BUFFER_STATE         LexBuffer;

/*******************************************************************************
 *
 * FUNCTION:    DtInitLexer, DtTerminateLexer
 *
 * PARAMETERS:  String              - Input string to be parsed
 *
 * RETURN:      None
 *
 * DESCRIPTION: Initialization and termination routines for lexer. Lexer needs
 *              a buffer to handle strings instead of a file.
 *
 ******************************************************************************/

int
DtInitLexer (
    char                    *String)
{

    LexBuffer = yy_scan_string (String);
    return (LexBuffer == NULL);
}

void
```

```
128 DtTerminateLexer (
129     void)
130 {
132     yy_delete_buffer (LexBuffer);
133 }
```

```
**********************************************************
   10045 Thu Dec 26 13:48:37 2013
new/usr/src/common/acpica/compiler/dtparser.y
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: dtparser.y - Bison input file for table compiler parser
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtparser")

int                         DtParserlex (void);
int                         DtParserparse (void);
void                        DtParsererror (char const *msg);
extern char                 *DtParsertext;
extern DT_FIELD             *Gbl_CurrentField;

UINT64                      DtParserResult; /* Expression return value */

/* Bison/yacc configuration */
```



```
#define yytname             DtParsername
#define YYDEBUG             1               /* Enable debug output */
#define YYERROR_VERBOSE     1               /* Verbose error messages */
#define YYFLAG              -32768

/* Define YYMALLOC/YYFREE to prevent redefinition errors  */

#define YYMALLOC            malloc
#define YYFREE              free
%}

%union
{
     UINT64                 value;
     UINT32                 op;
}

/*! [Begin] no source code translation */

%type  <value>  Expression

%token <op>     EXPOP_EOF
%token <op>     EXPOP_NEW_LINE
%token <op>     EXPOP_NUMBER
%token <op>     EXPOP_HEX_NUMBER
%token <op>     EXPOP_DECIMAL_NUMBER
%token <op>     EXPOP_LABEL
%token <op>     EXPOP_PAREN_OPEN
%token <op>     EXPOP_PAREN_CLOSE

%left <op>      EXPOP_LOGICAL_OR
%left <op>      EXPOP_LOGICAL_AND
%left <op>      EXPOP_OR
%left <op>      EXPOP_XOR
%left <op>      EXPOP_AND
%left <op>      EXPOP_EQUAL EXPOP_NOT_EQUAL
%left <op>      EXPOP_GREATER EXPOP_LESS EXPOP_GREATER_EQUAL EXPOP_LESS_EQUAL
%left <op>      EXPOP_SHIFT_RIGHT EXPOP_SHIFT_LEFT
%left <op>      EXPOP_ADD EXPOP_SUBTRACT
%left <op>      EXPOP_MULTIPLY EXPOP_DIVIDE EXPOP_MODULO
%right <op>     EXPOP_ONES_COMPLIMENT EXPOP_LOGICAL_NOT

%%

/*
 *  Operator precedence rules (from K&R)
 *
 *  1)      ( )
 *  2)      ! ~ (unary operators that are supported here)
 *  3)      *   /   %
 *  4)      +   -
 *  5)      >>  <<
 *  6)      <   >   <=  >=
 *  7)      ==  !=
 *  8)      &
 *  9)      ^
 *  10)     |
 *  11)     &&
 *  12)     ||
 */
Value
    : Expression EXPOP_NEW_LINE                     { DtParserResult=$1; return
    | Expression EXPOP_EOF                          { DtParserResult=$1; return
    ;

Expression
```

```
128       /* Unary operators */

130     : EXPOP_LOGICAL_NOT         Expression          { $$ = DtDoOperator ($2, EXP
131     | EXPOP_ONES_COMPLIMENT     Expression          { $$ = DtDoOperator ($2, EXP

133       /* Binary operators */

135     | Expression EXPOP_MULTIPLY         Expression  { $$ = DtDoOperator ($1, EXP
136     | Expression EXPOP_DIVIDE           Expression  { $$ = DtDoOperator ($1, EXP
137     | Expression EXPOP_MODULO           Expression  { $$ = DtDoOperator ($1, EXP
138     | Expression EXPOP_ADD              Expression  { $$ = DtDoOperator ($1, EXP
139     | Expression EXPOP_SUBTRACT         Expression  { $$ = DtDoOperator ($1, EXP
140     | Expression EXPOP_SHIFT_RIGHT      Expression  { $$ = DtDoOperator ($1, EXP
141     | Expression EXPOP_SHIFT_LEFT       Expression  { $$ = DtDoOperator ($1, EXP
142     | Expression EXPOP_GREATER          Expression  { $$ = DtDoOperator ($1, EXP
143     | Expression EXPOP_LESS             Expression  { $$ = DtDoOperator ($1, EXP
144     | Expression EXPOP_GREATER_EQUAL    Expression  { $$ = DtDoOperator ($1, EXP
145     | Expression EXPOP_LESS_EQUAL       Expression  { $$ = DtDoOperator ($1, EXP
146     | Expression EXPOP_EQUAL            Expression  { $$ = DtDoOperator ($1, EXP
147     | Expression EXPOP_NOT_EQUAL        Expression  { $$ = DtDoOperator ($1, EXP
148     | Expression EXPOP_AND              Expression  { $$ = DtDoOperator ($1, EXP
149     | Expression EXPOP_XOR              Expression  { $$ = DtDoOperator ($1, EXP
150     | Expression EXPOP_OR               Expression  { $$ = DtDoOperator ($1, EXP
151     | Expression EXPOP_LOGICAL_AND      Expression  { $$ = DtDoOperator ($1, EXP
152     | Expression EXPOP_LOGICAL_OR       Expression  { $$ = DtDoOperator ($1, EXP

154       /* Parentheses: '(' Expression ')' */

156     | EXPOP_PAREN_OPEN          Expression
157         EXPOP_PAREN_CLOSE                         { $$ = $2;}

159       /* Label references (prefixed with $) */

161     | EXPOP_LABEL                                 { $$ = DtResolveLabel (DtPar

163       /* Default base for a non-prefixed integer is 16 */

165     | EXPOP_NUMBER                                { UtStrtoul64 (DtParsertext,

167       /* Standard hex number (0x1234) */

169     | EXPOP_HEX_NUMBER                            { UtStrtoul64 (DtParsertext,

171       /* TBD: Decimal number with prefix (0d1234) - Not supported by UtStrtoul64

173     | EXPOP_DECIMAL_NUMBER                        { UtStrtoul64 (DtParsertext,
174     ;
175 %%

177 /*! [End] no source code translation !*/

179 /*
180  * Local support functions, including parser entry point
181  */
182 #define PR_FIRST_PARSE_OPCODE   EXPOP_EOF
183 #define PR_YYTNAME_START        3


186 /*******************************************************************************
187  *
188  * FUNCTION:    DtParsererror
189  *
190  * PARAMETERS:  Message             - Parser-generated error message
191  *
192  * RETURN:      None
```



```
193  *
194  * DESCRIPTION: Handler for parser errors
195  *
196  ******************************************************************************/

198 void
199 DtParsererror (
200     char const              *Message)
201 {
202     DtError (ASL_ERROR, ASL_MSG_SYNTAX,
203         Gbl_CurrentField, (char *) Message);
204 }


207 /*******************************************************************************
208  *
209  * FUNCTION:    DtGetOpName
210  *
211  * PARAMETERS:  ParseOpcode         - Parser token (EXPOP_*)
212  *
213  * RETURN:      Pointer to the opcode name
214  *
215  * DESCRIPTION: Get the ascii name of the parse opcode for debug output
216  *
217  ******************************************************************************/

219 char *
220 DtGetOpName (
221     UINT32                  ParseOpcode)
222 {
223 #ifdef ASL_YYTNAME_START
224     /*
225      * First entries (PR_YYTNAME_START) in yytname are special reserved names.
226      * Ignore first 6 characters of name (EXPOP_)
227      */
228     return ((char *) yytname
229         [(ParseOpcode - PR_FIRST_PARSE_OPCODE) + PR_YYTNAME_START] + 6);
230 #else
231     return ("[Unknown parser generator]");
232 #endif
233 }


236 /*******************************************************************************
237  *
238  * FUNCTION:    DtEvaluateExpression
239  *
240  * PARAMETERS:  ExprString          - Expression to be evaluated. Must be
241  *                                    terminated by either a newline or a NUL
242  *                                    string terminator
243  *
244  * RETURN:      64-bit value for the expression
245  *
246  * DESCRIPTION: Main entry point for the DT expression parser
247  *
248  ******************************************************************************/

250 UINT64
251 DtEvaluateExpression (
252     char                    *ExprString)
253 {

255     DbgPrint (ASL_DEBUG_OUTPUT,
256         "**** Input expression: %s  (Base 16)\n", ExprString);

258     /* Point lexer to the input string */
```

```
260     if (DtInitLexer (ExprString))
261     {
262         DtError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL,
263             Gbl_CurrentField, "Could not initialize lexer");
264         return (0);
265     }

267     /* Parse/Evaluate the input string (value returned in DtParserResult) */

269     DtParserparse ();
270     DtTerminateLexer ();

272     DbgPrint (ASL_DEBUG_OUTPUT,
273         "**** Parser returned value: %u (%8.8X%8.8X)\n",
274         (UINT32) DtParserResult, ACPI_FORMAT_UINT64 (DtParserResult));

276     return (DtParserResult);
277 }
```

```
**********************************************************
    9644 Thu Dec 26 13:48:38 2013
new/usr/src/common/acpica/compiler/dtsubtable.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dtsubtable.c - handling of subtables within ACPI tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTSUBTABLE_C__

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtsubtable")


/******************************************************************************
 *
 * FUNCTION:    DtCreateSubtable
 *
 * PARAMETERS:  Buffer              - Input buffer
 *              Length              - Buffer length
 *              RetSubtable         - Returned newly created subtable
 *
```



```
 * RETURN:      None
 *
 * DESCRIPTION: Create a subtable that is not listed with ACPI_DMTABLE_INFO
 *              For example, FACS has 24 bytes reserved at the end
 *              and it's not listed at AcpiDmTableInfoFacs
 *
 *****************************************************************************/

void
DtCreateSubtable (
    UINT8                   *Buffer,
    UINT32                  Length,
    DT_SUBTABLE             **RetSubtable)
{
    DT_SUBTABLE             *Subtable;


    Subtable = UtLocalCalloc (sizeof (DT_SUBTABLE));

    /* Create a new buffer for the subtable data */

    Subtable->Buffer = UtLocalCalloc (Length);
    ACPI_MEMCPY (Subtable->Buffer, Buffer, Length);

    Subtable->Length = Length;
    Subtable->TotalLength = Length;

    *RetSubtable = Subtable;
}


/******************************************************************************
 *
 * FUNCTION:    DtInsertSubtable
 *
 * PARAMETERS:  ParentTable         - The Parent of the new subtable
 *              Subtable            - The new subtable to insert
 *
 * RETURN:      None
 *
 * DESCRIPTION: Insert the new subtable to the parent table
 *
 *****************************************************************************/

void
DtInsertSubtable (
    DT_SUBTABLE             *ParentTable,
    DT_SUBTABLE             *Subtable)
{
    DT_SUBTABLE             *ChildTable;


    Subtable->Peer = NULL;
    Subtable->Parent = ParentTable;
    Subtable->Depth = ParentTable->Depth + 1;

    /* Link the new entry into the child list */

    if (!ParentTable->Child)
    {
        ParentTable->Child = Subtable;
    }
    else
    {
        /* Walk to the end of the child list */
```

```
127         ChildTable = ParentTable->Child;
128         while (ChildTable->Peer)
129         {
130             ChildTable = ChildTable->Peer;
131         }
132
133         /* Add new subtable at the end of the child list */
134
135         ChildTable->Peer = Subtable;
136     }
137 }


140 /*******************************************************************************
141  *
142  * FUNCTION:    DtPushSubtable
143  *
144  * PARAMETERS:  Subtable            - Subtable to push
145  *
146  * RETURN:      None
147  *
148  * DESCRIPTION: Push a subtable onto a subtable stack
149  *
150  ******************************************************************************/

152 void
153 DtPushSubtable (
154     DT_SUBTABLE             *Subtable)
155 {
156
157     Subtable->StackTop = Gbl_SubtableStack;
158     Gbl_SubtableStack = Subtable;
159 }


162 /*******************************************************************************
163  *
164  * FUNCTION:    DtPopSubtable
165  *
166  * PARAMETERS:  None
167  *
168  * RETURN:      None
169  *
170  * DESCRIPTION: Pop a subtable from a subtable stack. Uses global SubtableStack
171  *
172  ******************************************************************************/

174 void
175 DtPopSubtable (
176     void)
177 {
178     DT_SUBTABLE             *Subtable;
179
180
181     Subtable = Gbl_SubtableStack;
182
183     if (Subtable)
184     {
185         Gbl_SubtableStack = Subtable->StackTop;
186     }
187 }


190 /*******************************************************************************
191  *
192  * FUNCTION:    DtPeekSubtable
```



```
193  *
194  * PARAMETERS:  None
195  *
196  * RETURN:      The subtable on top of stack
197  *
198  * DESCRIPTION: Get the subtable on top of stack
199  *
200  ******************************************************************************/

202 DT_SUBTABLE *
203 DtPeekSubtable (
204     void)
205 {
206
207     return (Gbl_SubtableStack);
208 }


211 /*******************************************************************************
212  *
213  * FUNCTION:    DtGetNextSubtable
214  *
215  * PARAMETERS:  ParentTable         - Parent table whose children we are
216  *                                    getting
217  *              ChildTable          - Previous child that was found.
218  *                                    The NEXT child will be returned
219  *
220  * RETURN:      Pointer to the NEXT child or NULL if none is found.
221  *
222  * DESCRIPTION: Return the next peer subtable within the tree.
223  *
224  ******************************************************************************/

226 DT_SUBTABLE *
227 DtGetNextSubtable (
228     DT_SUBTABLE             *ParentTable,
229     DT_SUBTABLE             *ChildTable)
230 {
231     ACPI_FUNCTION_ENTRY ();


234     if (!ChildTable)
235     {
236         /* It's really the parent's _scope_ that we want */
237
238         return (ParentTable->Child);
239     }
240
241     /* Otherwise just return the next peer (NULL if at end-of-list) */
242
243     return (ChildTable->Peer);
244 }


247 /*******************************************************************************
248  *
249  * FUNCTION:    DtGetParentSubtable
250  *
251  * PARAMETERS:  Subtable            - Current subtable
252  *
253  * RETURN:      Parent of the given subtable
254  *
255  * DESCRIPTION: Get the parent of the given subtable in the tree
256  *
257  ******************************************************************************/
```

```
259 DT_SUBTABLE *
260 DtGetParentSubtable (
261     DT_SUBTABLE             *Subtable)
262 {

264     if (!Subtable)
265     {
266         return (NULL);
267     }

269     return (Subtable->Parent);
270 }


273 /******************************************************************************
274  *
275  * FUNCTION:    DtGetSubtableLength
276  *
277  * PARAMETERS:  Field               - Current field list pointer
278  *              Info                - Data table info
279  *
280  * RETURN:      Subtable length
281  *
282  * DESCRIPTION: Get length of bytes needed to compile the subtable
283  *
284  *****************************************************************************/

286 UINT32
287 DtGetSubtableLength (
288     DT_FIELD                *Field,
289     ACPI_DMTABLE_INFO       *Info)
290 {
291     UINT32                  ByteLength = 0;
292     UINT8                   Step;
293     UINT8                   i;


296     /* Walk entire Info table; Null name terminates */

298     for (; Info->Name; Info++)
299     {
300         if (Info->Opcode == ACPI_DMT_EXTRA_TEXT)
301         {
302             continue;
303         }

305         if (!Field)
306         {
307             goto Error;
308         }

310         ByteLength += DtGetFieldLength (Field, Info);

312         switch (Info->Opcode)
313         {
314         case ACPI_DMT_GAS:

316             Step = 5;
317             break;

319         case ACPI_DMT_HESTNTFY:

321             Step = 9;
322             break;

324         default:
```



```
326             Step = 1;
327             break;
328         }

330         for (i = 0; i < Step; i++)
331         {
332             if (!Field)
333             {
334                 goto Error;
335             }

337             Field = Field->Next;
338         }
339     }

341     return (ByteLength);

343 Error:
344     if (!Field)
345     {
346         sprintf (MsgBuffer, "Found NULL field - Field name \"%s\" needed",
347             Info->Name);
348         DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, MsgBuffer);
349     }

351     return (ASL_EOF);
352 }


355 /******************************************************************************
356  *
357  * FUNCTION:    DtSetSubtableLength
358  *
359  * PARAMETERS:  Subtable            - Subtable
360  *
361  * RETURN:      None
362  *
363  * DESCRIPTION: Set length of the subtable into its length field
364  *
365  *****************************************************************************/

367 void
368 DtSetSubtableLength (
369     DT_SUBTABLE             *Subtable)
370 {

372     if (!Subtable->LengthField)
373     {
374         return;
375     }

377     ACPI_MEMCPY (Subtable->LengthField, &Subtable->TotalLength,
378         Subtable->SizeOfLengthField);
379 }
```

```
**********************************************************
   64596 Thu Dec 26 13:48:38 2013
new/usr/src/common/acpica/compiler/dttable.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dttable.c - handling for specific ACPI tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTTABLE_C__

/* Compile all complex data tables */

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dttable")


/* TBD: merge these into dmtbinfo.c? */

static ACPI_DMTABLE_INFO           TableInfoAsfAddress[] =
{
    {ACPI_DMT_BUFFER,   0,              "Addresses", 0},
```



```
    {ACPI_DMT_EXIT,     0,              NULL, 0}
};

static ACPI_DMTABLE_INFO           TableInfoDmarPciPath[] =
{
    {ACPI_DMT_PCI_PATH, 0,              "PCI Path", 0},
    {ACPI_DMT_EXIT,     0,              NULL, 0}
};


/* TBD: move to acmacros.h */

#define ACPI_SUB_PTR(t, a, b) \
    ACPI_CAST_PTR (t, (ACPI_CAST_PTR (UINT8, (a)) - (ACPI_SIZE)(b)))


/* Local prototypes */

static ACPI_STATUS
DtCompileTwoSubtables (
    void                    **List,
    ACPI_DMTABLE_INFO       *TableInfo1,
    ACPI_DMTABLE_INFO       *TableInfo2);


/******************************************************************************
 *
 * FUNCTION:    DtCompileTwoSubtables
 *
 * PARAMETERS:  List                - Current field list pointer
 *              TableInfo1          - Info table 1
 *              TableInfo1          - Info table 2
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile tables with a header and one or more same subtables.
 *              Include CPEP, EINJ, ERST, MCFG, MSCT, WDAT
 *
 *****************************************************************************/

static ACPI_STATUS
DtCompileTwoSubtables (
    void                    **List,
    ACPI_DMTABLE_INFO       *TableInfo1,
    ACPI_DMTABLE_INFO       *TableInfo2)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;


    Status = DtCompileTable (PFieldList, TableInfo1, &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);

    while (*PFieldList)
    {
        Status = DtCompileTable (PFieldList, TableInfo2, &Subtable, FALSE);
        if (ACPI_FAILURE (Status))
        {
```

```
126             return (Status);
127         }
129         DtInsertSubtable (ParentTable, Subtable);
130     }
132     return (AE_OK);
133 }
136 /******************************************************************************
137  *
138  * FUNCTION:    DtCompileFacs
139  *
140  * PARAMETERS:  PFieldList          - Current field list pointer
141  *
142  * RETURN:      Status
143  *
144  * DESCRIPTION: Compile FACS.
145  *
146  *****************************************************************************/
148 ACPI_STATUS
149 DtCompileFacs (
150     DT_FIELD                **PFieldList)
151 {
152     DT_SUBTABLE             *Subtable;
153     UINT8                   *ReservedBuffer;
154     ACPI_STATUS             Status;
155     UINT32                  ReservedSize;
158     Status = DtCompileTable (PFieldList, AcpiDmTableInfoFacs,
159                 &Gbl_RootTable, TRUE);
160     if (ACPI_FAILURE (Status))
161     {
162         return (Status);
163     }
165     /* Large FACS reserved area at the end of the table */
167     ReservedSize = (UINT32) sizeof (((ACPI_TABLE_FACS *) NULL)->Reserved1);
168     ReservedBuffer = UtLocalCalloc (ReservedSize);
170     DtCreateSubtable (ReservedBuffer, ReservedSize, &Subtable);
172     ACPI_FREE (ReservedBuffer);
173     DtInsertSubtable (Gbl_RootTable, Subtable);
174     return (AE_OK);
175 }
178 /******************************************************************************
179  *
180  * FUNCTION:    DtCompileRsdp
181  *
182  * PARAMETERS:  PFieldList          - Current field list pointer
183  *
184  * RETURN:      Status
185  *
186  * DESCRIPTION: Compile RSDP.
187  *
188  *****************************************************************************/
190 ACPI_STATUS
191 DtCompileRsdp (
```



```
192     DT_FIELD                **PFieldList)
193 {
194     DT_SUBTABLE             *Subtable;
195     ACPI_TABLE_RSDP         *Rsdp;
196     ACPI_RSDP_EXTENSION     *RsdpExtension;
197     ACPI_STATUS             Status;
200     /* Compile the "common" RSDP (ACPI 1.0) */
202     Status = DtCompileTable (PFieldList, AcpiDmTableInfoRsdp1,
203                 &Gbl_RootTable, TRUE);
204     if (ACPI_FAILURE (Status))
205     {
206         return (Status);
207     }
209     Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, Gbl_RootTable->Buffer);
210     DtSetTableChecksum (&Rsdp->Checksum);
212     if (Rsdp->Revision > 0)
213     {
214         /* Compile the "extended" part of the RSDP as a subtable */
216         Status = DtCompileTable (PFieldList, AcpiDmTableInfoRsdp2,
217                     &Subtable, TRUE);
218         if (ACPI_FAILURE (Status))
219         {
220             return (Status);
221         }
223         DtInsertSubtable (Gbl_RootTable, Subtable);
225         /* Set length and extended checksum for entire RSDP */
227         RsdpExtension = ACPI_CAST_PTR (ACPI_RSDP_EXTENSION, Subtable->Buffer);
228         RsdpExtension->Length = Gbl_RootTable->Length + Subtable->Length;
229         DtSetTableChecksum (&RsdpExtension->ExtendedChecksum);
230     }
232     return (AE_OK);
233 }
236 /******************************************************************************
237  *
238  * FUNCTION:    DtCompileAsf
239  *
240  * PARAMETERS:  List                - Current field list pointer
241  *
242  * RETURN:      Status
243  *
244  * DESCRIPTION: Compile ASF!.
245  *
246  *****************************************************************************/
248 ACPI_STATUS
249 DtCompileAsf (
250     void                    **List)
251 {
252     ACPI_ASF_INFO           *AsfTable;
253     DT_SUBTABLE             *Subtable;
254     DT_SUBTABLE             *ParentTable;
255     ACPI_DMTABLE_INFO       *InfoTable;
256     ACPI_DMTABLE_INFO       *DataInfoTable = NULL;
257     UINT32                  DataCount = 0;
```

```
258     ACPI_STATUS             Status;
259     UINT32                  i;
260     DT_FIELD                **PFieldList = (DT_FIELD **) List;
261     DT_FIELD                *SubtableStart;


264     while (*PFieldList)
265     {
266         SubtableStart = *PFieldList;
267         Status = DtCompileTable (PFieldList, AcpiDmTableInfoAsfHdr,
268                     &Subtable, TRUE);
269         if (ACPI_FAILURE (Status))
270         {
271             return (Status);
272         }

274         ParentTable = DtPeekSubtable ();
275         DtInsertSubtable (ParentTable, Subtable);
276         DtPushSubtable (Subtable);

278         AsfTable = ACPI_CAST_PTR (ACPI_ASF_INFO, Subtable->Buffer);

280         switch (AsfTable->Header.Type & 0x7F) /* Mask off top bit */
281         {
282         case ACPI_ASF_TYPE_INFO:

284             InfoTable = AcpiDmTableInfoAsf0;
285             break;

287         case ACPI_ASF_TYPE_ALERT:

289             InfoTable = AcpiDmTableInfoAsf1;
290             break;

292         case ACPI_ASF_TYPE_CONTROL:

294             InfoTable = AcpiDmTableInfoAsf2;
295             break;

297         case ACPI_ASF_TYPE_BOOT:

299             InfoTable = AcpiDmTableInfoAsf3;
300             break;

302         case ACPI_ASF_TYPE_ADDRESS:

304             InfoTable = AcpiDmTableInfoAsf4;
305             break;

307         default:

309             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "ASF!");
310             return (AE_ERROR);
311         }

313         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
314         if (ACPI_FAILURE (Status))
315         {
316             return (Status);
317         }

319         ParentTable = DtPeekSubtable ();
320         DtInsertSubtable (ParentTable, Subtable);

322         switch (AsfTable->Header.Type & 0x7F) /* Mask off top bit */
323         {
```



```
324         case ACPI_ASF_TYPE_INFO:

326             DataInfoTable = NULL;
327             break;

329         case ACPI_ASF_TYPE_ALERT:

331             DataInfoTable = AcpiDmTableInfoAsf1a;
332             DataCount = ACPI_CAST_PTR (ACPI_ASF_ALERT,
333                         ACPI_SUB_PTR (UINT8, Subtable->Buffer,
334                             sizeof (ACPI_ASF_HEADER)))->Alerts;
335             break;

337         case ACPI_ASF_TYPE_CONTROL:

339             DataInfoTable = AcpiDmTableInfoAsf2a;
340             DataCount = ACPI_CAST_PTR (ACPI_ASF_REMOTE,
341                         ACPI_SUB_PTR (UINT8, Subtable->Buffer,
342                             sizeof (ACPI_ASF_HEADER)))->Controls;
343             break;

345         case ACPI_ASF_TYPE_BOOT:

347             DataInfoTable = NULL;
348             break;

350         case ACPI_ASF_TYPE_ADDRESS:

352             DataInfoTable = TableInfoAsfAddress;
353             DataCount = ACPI_CAST_PTR (ACPI_ASF_ADDRESS,
354                         ACPI_SUB_PTR (UINT8, Subtable->Buffer,
355                             sizeof (ACPI_ASF_HEADER)))->Devices;
356             break;

358         default:

360             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "ASF!");
361             return (AE_ERROR);
362         }

364         if (DataInfoTable)
365         {
366             switch (AsfTable->Header.Type & 0x7F)
367             {
368             case ACPI_ASF_TYPE_ADDRESS:

370                 while (DataCount > 0)
371                 {
372                     Status = DtCompileTable (PFieldList, DataInfoTable,
373                                 &Subtable, TRUE);
374                     if (ACPI_FAILURE (Status))
375                     {
376                         return (Status);
377                     }

379                     DtInsertSubtable (ParentTable, Subtable);
380                     DataCount = DataCount - Subtable->Length;
381                 }
382                 break;

384             default:

386                 for (i = 0; i < DataCount; i++)
387                 {
388                     Status = DtCompileTable (PFieldList, DataInfoTable,
389                                 &Subtable, TRUE);
```

```
                    if (ACPI_FAILURE (Status))
                    {
                        return (Status);
                    }

                    DtInsertSubtable (ParentTable, Subtable);
                }
                break;
            }
        }

        DtPopSubtable ();
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileCpep
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile CPEP.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileCpep (
    void                    **List)
{
    ACPI_STATUS             Status;


    Status = DtCompileTwoSubtables (List,
                AcpiDmTableInfoCpep, AcpiDmTableInfoCpep0);
    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileCsrt
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile CSRT.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileCsrt (
    void                    **List)
{
    ACPI_STATUS             Status = AE_OK;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    UINT32                  DescriptorCount;
    UINT32                  GroupLength;
```



```
    /* Sub-tables (Resource Groups) */

    while (*PFieldList)
    {
        /* Resource group subtable */

        Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt0,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        /* Compute the number of resource descriptors */

        GroupLength =
            (ACPI_CAST_PTR (ACPI_CSRT_GROUP,
                Subtable->Buffer))->Length -
            (ACPI_CAST_PTR (ACPI_CSRT_GROUP,
                Subtable->Buffer))->SharedInfoLength -
            sizeof (ACPI_CSRT_GROUP);

        DescriptorCount = (GroupLength  /
            sizeof (ACPI_CSRT_DESCRIPTOR));

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        /* Shared info subtable (One per resource group) */

        Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt1,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);

        /* Sub-Subtables (Resource Descriptors) */

        while (*PFieldList && DescriptorCount)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt2,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            ParentTable = DtPeekSubtable ();
            DtInsertSubtable (ParentTable, Subtable);
            DescriptorCount--;
        }

        DtPopSubtable ();
    }

    return (Status);
}


/******************************************************************************
```

```
 *
 * FUNCTION:    DtCompileDbg2
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile DBG2.
 *
 ******************************************************************************/

ACPI_STATUS
DtCompileDbg2 (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    UINT32                  SubtableCount;
    ACPI_DBG2_HEADER        *Dbg2Header;
    ACPI_DBG2_DEVICE        *DeviceInfo;
    UINT16                  CurrentOffset;
    UINT32                  i;


    /* Main table */

    Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2, &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);

    /* Main table fields */

    Dbg2Header = ACPI_CAST_PTR (ACPI_DBG2_HEADER, Subtable->Buffer);
    Dbg2Header->InfoOffset = sizeof (ACPI_TABLE_HEADER) + ACPI_PTR_DIFF (
        ACPI_ADD_PTR (UINT8, Dbg2Header, sizeof (ACPI_DBG2_HEADER)), Dbg2Header)

    SubtableCount = Dbg2Header->InfoCount;
    DtPushSubtable (Subtable);

    /* Process all Device Information subtables (Count = InfoCount) */

    while (*PFieldList && SubtableCount)
    {
        /* Subtable: Debug Device Information */

        Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Device,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        DeviceInfo = ACPI_CAST_PTR (ACPI_DBG2_DEVICE, Subtable->Buffer);
        CurrentOffset = (UINT16) sizeof (ACPI_DBG2_DEVICE);

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);
```



```
        ParentTable = DtPeekSubtable ();

        /* BaseAddressRegister GAS array (Required, size is RegisterCount) */

        DeviceInfo->BaseAddressOffset = CurrentOffset;
        for (i = 0; *PFieldList && (i < DeviceInfo->RegisterCount); i++)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Addr,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            CurrentOffset += (UINT16) sizeof (ACPI_GENERIC_ADDRESS);
            DtInsertSubtable (ParentTable, Subtable);
        }

        /* AddressSize array (Required, size = RegisterCount) */

        DeviceInfo->AddressSizeOffset = CurrentOffset;
        for (i = 0; *PFieldList && (i < DeviceInfo->RegisterCount); i++)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Size,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            CurrentOffset += (UINT16) sizeof (UINT32);
            DtInsertSubtable (ParentTable, Subtable);
        }

        /* NamespaceString device identifier (Required, size = NamePathLength) *

        DeviceInfo->NamepathOffset = CurrentOffset;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Name,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        /* Update the device info header */

        DeviceInfo->NamepathLength = (UINT16) Subtable->Length;
        CurrentOffset += (UINT16) DeviceInfo->NamepathLength;
        DtInsertSubtable (ParentTable, Subtable);

        /* OemData - Variable-length data (Optional, size = OemDataLength) */

        Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2OemData,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        /* Update the device info header (zeros if no OEM data present) */

        DeviceInfo->OemDataOffset = 0;
        DeviceInfo->OemDataLength = 0;

        /* Optional subtable (OemData) */
```

```
654         if (Subtable && Subtable->Length)
655         {
656             DeviceInfo->OemDataOffset = CurrentOffset;
657             DeviceInfo->OemDataLength = (UINT16) Subtable->Length;

659             DtInsertSubtable (ParentTable, Subtable);
660         }

662         SubtableCount--;
663         DtPopSubtable (); /* Get next Device Information subtable */
664     }

666     DtPopSubtable ();
667     return (AE_OK);
668 }


671 /******************************************************************************
672  *
673  * FUNCTION:    DtCompileDmar
674  *
675  * PARAMETERS:  List                - Current field list pointer
676  *
677  * RETURN:      Status
678  *
679  * DESCRIPTION: Compile DMAR.
680  *
681  *****************************************************************************/

683 ACPI_STATUS
684 DtCompileDmar (
685     void                    **List)
686 {
687     ACPI_STATUS             Status;
688     DT_SUBTABLE             *Subtable;
689     DT_SUBTABLE             *ParentTable;
690     DT_FIELD                **PFieldList = (DT_FIELD **) List;
691     DT_FIELD                *SubtableStart;
692     ACPI_DMTABLE_INFO       *InfoTable;
693     ACPI_DMAR_HEADER        *DmarHeader;
694     ACPI_DMAR_DEVICE_SCOPE  *DmarDeviceScope;
695     UINT32                  DeviceScopeLength;
696     UINT32                  PciPathLength;


699     Status = DtCompileTable (PFieldList, AcpiDmTableInfoDmar, &Subtable, TRUE);
700     if (ACPI_FAILURE (Status))
701     {
702         return (Status);
703     }

705     ParentTable = DtPeekSubtable ();
706     DtInsertSubtable (ParentTable, Subtable);
707     DtPushSubtable (Subtable);

709     while (*PFieldList)
710     {
711         /* DMAR Header */

713         SubtableStart = *PFieldList;
714         Status = DtCompileTable (PFieldList, AcpiDmTableInfoDmarHdr,
715                     &Subtable, TRUE);
716         if (ACPI_FAILURE (Status))
717         {
718             return (Status);
719         }
```



```
721         ParentTable = DtPeekSubtable ();
722         DtInsertSubtable (ParentTable, Subtable);
723         DtPushSubtable (Subtable);

725         DmarHeader = ACPI_CAST_PTR (ACPI_DMAR_HEADER, Subtable->Buffer);

727         switch (DmarHeader->Type)
728         {
729         case ACPI_DMAR_TYPE_HARDWARE_UNIT:

731             InfoTable = AcpiDmTableInfoDmar0;
732             break;

734         case ACPI_DMAR_TYPE_RESERVED_MEMORY:

736             InfoTable = AcpiDmTableInfoDmar1;
737             break;

739         case ACPI_DMAR_TYPE_ATSR:

741             InfoTable = AcpiDmTableInfoDmar2;
742             break;

744         case ACPI_DMAR_HARDWARE_AFFINITY:

746             InfoTable = AcpiDmTableInfoDmar3;
747             break;

749         default:

751             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "DMAR");
752             return (AE_ERROR);
753         }

755         /* DMAR Subtable */

757         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
758         if (ACPI_FAILURE (Status))
759         {
760             return (Status);
761         }

763         ParentTable = DtPeekSubtable ();
764         DtInsertSubtable (ParentTable, Subtable);
765         DtPushSubtable (Subtable);

767         /* Optional Device Scope subtables */

769         DeviceScopeLength = DmarHeader->Length - Subtable->Length -
770             ParentTable->Length;
771         while (DeviceScopeLength)
772         {
773             Status = DtCompileTable (PFieldList, AcpiDmTableInfoDmarScope,
774                         &Subtable, FALSE);
775             if (Status == AE_NOT_FOUND)
776             {
777                 break;
778             }

780             ParentTable = DtPeekSubtable ();
781             DtInsertSubtable (ParentTable, Subtable);
782             DtPushSubtable (Subtable);

784             DmarDeviceScope = ACPI_CAST_PTR (ACPI_DMAR_DEVICE_SCOPE, Subtable->B
```

```
786             /* Optional PCI Paths */
788             PciPathLength = DmarDeviceScope->Length - Subtable->Length;
789             while (PciPathLength)
790             {
791                 Status = DtCompileTable (PFieldList, TableInfoDmarPciPath,
792                             &Subtable, FALSE);
793                 if (Status == AE_NOT_FOUND)
794                 {
795                     DtPopSubtable ();
796                     break;
797                 }
799                 ParentTable = DtPeekSubtable ();
800                 DtInsertSubtable (ParentTable, Subtable);
801                 PciPathLength -= Subtable->Length;
802             }
804             DtPopSubtable ();
805             DeviceScopeLength -= DmarDeviceScope->Length;
806         }
808         DtPopSubtable ();
809         DtPopSubtable ();
810     }
812     return (AE_OK);
813 }
816 /******************************************************************************
817  *
818  * FUNCTION:    DtCompileEinj
819  *
820  * PARAMETERS:  List                - Current field list pointer
821  *
822  * RETURN:      Status
823  *
824  * DESCRIPTION: Compile EINJ.
825  *
826  *****************************************************************************/
828 ACPI_STATUS
829 DtCompileEinj (
830     void                    **List)
831 {
832     ACPI_STATUS             Status;
835     Status = DtCompileTwoSubtables (List,
836                 AcpiDmTableInfoEinj, AcpiDmTableInfoEinj0);
837     return (Status);
838 }
841 /******************************************************************************
842  *
843  * FUNCTION:    DtCompileErst
844  *
845  * PARAMETERS:  List                - Current field list pointer
846  *
847  * RETURN:      Status
848  *
849  * DESCRIPTION: Compile ERST.
850  *
851  *****************************************************************************/
```



```
853 ACPI_STATUS
854 DtCompileErst (
855     void                    **List)
856 {
857     ACPI_STATUS             Status;
860     Status = DtCompileTwoSubtables (List,
861                 AcpiDmTableInfoErst, AcpiDmTableInfoEinj0);
862     return (Status);
863 }
866 /******************************************************************************
867  *
868  * FUNCTION:    DtCompileFadt
869  *
870  * PARAMETERS:  List                - Current field list pointer
871  *
872  * RETURN:      Status
873  *
874  * DESCRIPTION: Compile FADT.
875  *
876  *****************************************************************************/
878 ACPI_STATUS
879 DtCompileFadt (
880     void                    **List)
881 {
882     ACPI_STATUS             Status;
883     DT_SUBTABLE             *Subtable;
884     DT_SUBTABLE             *ParentTable;
885     DT_FIELD                **PFieldList = (DT_FIELD **) List;
886     ACPI_TABLE_HEADER       *Table;
887     UINT8                   Revision;
890     Status = DtCompileTable (PFieldList, AcpiDmTableInfoFadt1,
891                 &Subtable, TRUE);
892     if (ACPI_FAILURE (Status))
893     {
894         return (Status);
895     }
897     ParentTable = DtPeekSubtable ();
898     DtInsertSubtable (ParentTable, Subtable);
900     Table = ACPI_CAST_PTR (ACPI_TABLE_HEADER, ParentTable->Buffer);
901     Revision = Table->Revision;
903     if (Revision == 2)
904     {
905         Status = DtCompileTable (PFieldList, AcpiDmTableInfoFadt2,
906                     &Subtable, TRUE);
907         if (ACPI_FAILURE (Status))
908         {
909             return (Status);
910         }
912         DtInsertSubtable (ParentTable, Subtable);
913     }
914     else if (Revision >= 2)
915     {
916         Status = DtCompileTable (PFieldList, AcpiDmTableInfoFadt3,
917                     &Subtable, TRUE);
```

```
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        DtInsertSubtable (ParentTable, Subtable);

        if (Revision >= 5)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoFadt5,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            DtInsertSubtable (ParentTable, Subtable);
        }
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileFpdt
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile FPDT.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileFpdt (
    void                    **List)
{
    ACPI_STATUS             Status;
    ACPI_FPDT_HEADER        *FpdtHeader;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    ACPI_DMTABLE_INFO       *InfoTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    DT_FIELD                *SubtableStart;


    while (*PFieldList)
    {
        SubtableStart = *PFieldList;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoFpdtHdr,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        FpdtHeader = ACPI_CAST_PTR (ACPI_FPDT_HEADER, Subtable->Buffer);

        switch (FpdtHeader->Type)
```



```
        {
        case ACPI_FPDT_TYPE_BOOT:

            InfoTable = AcpiDmTableInfoFpdt0;
            break;

        case ACPI_FPDT_TYPE_S3PERF:

            InfoTable = AcpiDmTableInfoFpdt1;
            break;

        default:

            DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "FPDT");
            return (AE_ERROR);
            break;
        }

        Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPopSubtable ();
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileHest
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile HEST.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileHest (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    DT_FIELD                *SubtableStart;
    ACPI_DMTABLE_INFO       *InfoTable;
    UINT16                  Type;
    UINT32                  BankCount;


    Status = DtCompileTable (PFieldList, AcpiDmTableInfoHest,
                &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }
```

```
1050     ParentTable = DtPeekSubtable ();
1051     DtInsertSubtable (ParentTable, Subtable);
1052
1053     while (*PFieldList)
1054     {
1055         /* Get subtable type */
1056
1057         SubtableStart = *PFieldList;
1058         DtCompileInteger ((UINT8 *) &Type, *PFieldList, 2, 0);
1059
1060         switch (Type)
1061         {
1062         case ACPI_HEST_TYPE_IA32_CHECK:
1063
1064             InfoTable = AcpiDmTableInfoHest0;
1065             break;
1066
1067         case ACPI_HEST_TYPE_IA32_CORRECTED_CHECK:
1068
1069             InfoTable = AcpiDmTableInfoHest1;
1070             break;
1071
1072         case ACPI_HEST_TYPE_IA32_NMI:
1073
1074             InfoTable = AcpiDmTableInfoHest2;
1075             break;
1076
1077         case ACPI_HEST_TYPE_AER_ROOT_PORT:
1078
1079             InfoTable = AcpiDmTableInfoHest6;
1080             break;
1081
1082         case ACPI_HEST_TYPE_AER_ENDPOINT:
1083
1084             InfoTable = AcpiDmTableInfoHest7;
1085             break;
1086
1087         case ACPI_HEST_TYPE_AER_BRIDGE:
1088
1089             InfoTable = AcpiDmTableInfoHest8;
1090             break;
1091
1092         case ACPI_HEST_TYPE_GENERIC_ERROR:
1093
1094             InfoTable = AcpiDmTableInfoHest9;
1095             break;
1096
1097         default:
1098
1099             /* Cannot continue on unknown type */
1100
1101             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "HEST");
1102             return (AE_ERROR);
1103         }
1104
1105         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1106         if (ACPI_FAILURE (Status))
1107         {
1108             return (Status);
1109         }
1110
1111         DtInsertSubtable (ParentTable, Subtable);
1112
1113         /*
1114          * Additional subtable data - IA32 Error Bank(s)
1115          */
```



```
1116         BankCount = 0;
1117         switch (Type)
1118         {
1119         case ACPI_HEST_TYPE_IA32_CHECK:
1120
1121             BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_MACHINE_CHECK,
1122                         Subtable->Buffer))->NumHardwareBanks;
1123             break;
1124
1125         case ACPI_HEST_TYPE_IA32_CORRECTED_CHECK:
1126
1127             BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_CORRECTED,
1128                         Subtable->Buffer))->NumHardwareBanks;
1129             break;
1130
1131         default:
1132
1133             break;
1134         }
1135
1136         while (BankCount)
1137         {
1138             Status = DtCompileTable (PFieldList, AcpiDmTableInfoHestBank,
1139                         &Subtable, TRUE);
1140             if (ACPI_FAILURE (Status))
1141             {
1142                 return (Status);
1143             }
1144
1145             DtInsertSubtable (ParentTable, Subtable);
1146             BankCount--;
1147         }
1148     }
1149
1150     return (AE_OK);
1151 }
1152
1153
1154 /******************************************************************************
1155  *
1156  * FUNCTION:    DtCompileIvrs
1157  *
1158  * PARAMETERS:  List                - Current field list pointer
1159  *
1160  * RETURN:      Status
1161  *
1162  * DESCRIPTION: Compile IVRS.
1163  *
1164  *****************************************************************************/
1165
1166 ACPI_STATUS
1167 DtCompileIvrs (
1168     void                    **List)
1169 {
1170     ACPI_STATUS             Status;
1171     DT_SUBTABLE             *Subtable;
1172     DT_SUBTABLE             *ParentTable;
1173     DT_FIELD                **PFieldList = (DT_FIELD **) List;
1174     DT_FIELD                *SubtableStart;
1175     ACPI_DMTABLE_INFO       *InfoTable;
1176     ACPI_IVRS_HEADER        *IvrsHeader;
1177     UINT8                   EntryType;
1178
1179
1180     Status = DtCompileTable (PFieldList, AcpiDmTableInfoIvrs,
1181                 &Subtable, TRUE);
```

```
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);

    while (*PFieldList)
    {
        SubtableStart = *PFieldList;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoIvrsHdr,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        IvrsHeader = ACPI_CAST_PTR (ACPI_IVRS_HEADER, Subtable->Buffer);

        switch (IvrsHeader->Type)
        {
        case ACPI_IVRS_TYPE_HARDWARE:

            InfoTable = AcpiDmTableInfoIvrs0;
            break;

        case ACPI_IVRS_TYPE_MEMORY1:
        case ACPI_IVRS_TYPE_MEMORY2:
        case ACPI_IVRS_TYPE_MEMORY3:

            InfoTable = AcpiDmTableInfoIvrs1;
            break;

        default:

            DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "IVRS");
            return (AE_ERROR);
        }

        Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);

        if (IvrsHeader->Type == ACPI_IVRS_TYPE_HARDWARE)
        {
            while (*PFieldList &&
                    !ACPI_STRCMP ((*PFieldList)->Name, "Entry Type"))
            {
                SubtableStart = *PFieldList;
                DtCompileInteger (&EntryType, *PFieldList, 1, 0);

                switch (EntryType)
                {
                /* 4-byte device entries */

                case ACPI_IVRS_TYPE_PAD4:
```



```
                case ACPI_IVRS_TYPE_ALL:
                case ACPI_IVRS_TYPE_SELECT:
                case ACPI_IVRS_TYPE_START:
                case ACPI_IVRS_TYPE_END:

                    InfoTable = AcpiDmTableInfoIvrs4;
                    break;

                /* 8-byte entries, type A */

                case ACPI_IVRS_TYPE_ALIAS_SELECT:
                case ACPI_IVRS_TYPE_ALIAS_START:

                    InfoTable = AcpiDmTableInfoIvrs8a;
                    break;

                /* 8-byte entries, type B */

                case ACPI_IVRS_TYPE_PAD8:
                case ACPI_IVRS_TYPE_EXT_SELECT:
                case ACPI_IVRS_TYPE_EXT_START:

                    InfoTable = AcpiDmTableInfoIvrs8b;
                    break;

                /* 8-byte entries, type C */

                case ACPI_IVRS_TYPE_SPECIAL:

                    InfoTable = AcpiDmTableInfoIvrs8c;
                    break;

                default:

                    DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart,
                        "IVRS Device Entry");
                    return (AE_ERROR);
                }

                Status = DtCompileTable (PFieldList, InfoTable,
                            &Subtable, TRUE);
                if (ACPI_FAILURE (Status))
                {
                    return (Status);
                }

                DtInsertSubtable (ParentTable, Subtable);
            }
        }

        DtPopSubtable ();
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileMadt
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile MADT.
```

```
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileMadt (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    DT_FIELD                *SubtableStart;
    ACPI_SUBTABLE_HEADER    *MadtHeader;
    ACPI_DMTABLE_INFO       *InfoTable;


    Status = DtCompileTable (PFieldList, AcpiDmTableInfoMadt,
                &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);

    while (*PFieldList)
    {
        SubtableStart = *PFieldList;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoMadtHdr,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        MadtHeader = ACPI_CAST_PTR (ACPI_SUBTABLE_HEADER, Subtable->Buffer);

        switch (MadtHeader->Type)
        {
        case ACPI_MADT_TYPE_LOCAL_APIC:

            InfoTable = AcpiDmTableInfoMadt0;
            break;

        case ACPI_MADT_TYPE_IO_APIC:

            InfoTable = AcpiDmTableInfoMadt1;
            break;

        case ACPI_MADT_TYPE_INTERRUPT_OVERRIDE:

            InfoTable = AcpiDmTableInfoMadt2;
            break;

        case ACPI_MADT_TYPE_NMI_SOURCE:

            InfoTable = AcpiDmTableInfoMadt3;
            break;

        case ACPI_MADT_TYPE_LOCAL_APIC_NMI:
```



```
            InfoTable = AcpiDmTableInfoMadt4;
            break;

        case ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE:

            InfoTable = AcpiDmTableInfoMadt5;
            break;

        case ACPI_MADT_TYPE_IO_SAPIC:

            InfoTable = AcpiDmTableInfoMadt6;
            break;

        case ACPI_MADT_TYPE_LOCAL_SAPIC:

            InfoTable = AcpiDmTableInfoMadt7;
            break;

        case ACPI_MADT_TYPE_INTERRUPT_SOURCE:

            InfoTable = AcpiDmTableInfoMadt8;
            break;

        case ACPI_MADT_TYPE_LOCAL_X2APIC:

            InfoTable = AcpiDmTableInfoMadt9;
            break;

        case ACPI_MADT_TYPE_LOCAL_X2APIC_NMI:

            InfoTable = AcpiDmTableInfoMadt10;
            break;

        case ACPI_MADT_TYPE_GENERIC_INTERRUPT:

            InfoTable = AcpiDmTableInfoMadt11;
            break;

        case ACPI_MADT_TYPE_GENERIC_DISTRIBUTOR:

            InfoTable = AcpiDmTableInfoMadt12;
            break;

        default:

            DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "MADT");
            return (AE_ERROR);
        }

        Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPopSubtable ();
    }

    return (AE_OK);
}


/*****************************************************************************
 *
```

```
 * FUNCTION:    DtCompileMcfg
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile MCFG.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileMcfg (
    void                    **List)
{
    ACPI_STATUS             Status;


    Status = DtCompileTwoSubtables (List,
                AcpiDmTableInfoMcfg, AcpiDmTableInfoMcfg0);
    return (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    DtCompileMpst
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile MPST.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileMpst (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    ACPI_MPST_CHANNEL       *MpstChannelInfo;
    ACPI_MPST_POWER_NODE    *MpstPowerNode;
    ACPI_MPST_DATA_HDR      *MpstDataHeader;
    UINT16                  SubtableCount;
    UINT32                  PowerStateCount;
    UINT32                  ComponentCount;


    /* Main table */

    Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst, &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);
    DtPushSubtable (Subtable);

    MpstChannelInfo = ACPI_CAST_PTR (ACPI_MPST_CHANNEL, Subtable->Buffer);
    SubtableCount = MpstChannelInfo->PowerNodeCount;
```



```
    while (*PFieldList && SubtableCount)
    {
        /* Subtable: Memory Power Node(s) */

        Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        MpstPowerNode = ACPI_CAST_PTR (ACPI_MPST_POWER_NODE, Subtable->Buffer);
        PowerStateCount = MpstPowerNode->NumPowerStates;
        ComponentCount = MpstPowerNode->NumPhysicalComponents;

        ParentTable = DtPeekSubtable ();

        /* Sub-subtables - Memory Power State Structure(s) */

        while (*PFieldList && PowerStateCount)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0A,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            DtInsertSubtable (ParentTable, Subtable);
            PowerStateCount--;
        }

        /* Sub-subtables - Physical Component ID Structure(s) */

        while (*PFieldList && ComponentCount)
        {
            Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0B,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            DtInsertSubtable (ParentTable, Subtable);
            ComponentCount--;
        }

        SubtableCount--;
        DtPopSubtable ();
    }

    /* Subtable: Count of Memory Power State Characteristic structures */

    DtPopSubtable ();

    Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst1, &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
```

```
1578     DtInsertSubtable (ParentTable, Subtable);
1579     DtPushSubtable (Subtable);

1581     MpstDataHeader = ACPI_CAST_PTR (ACPI_MPST_DATA_HDR, Subtable->Buffer);
1582     SubtableCount = MpstDataHeader->CharacteristicsCount;

1584     ParentTable = DtPeekSubtable ();

1586     /* Subtable: Memory Power State Characteristics structure(s) */

1588     while (*PFieldList && SubtableCount)
1589     {
1590         Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst2,
1591                     &Subtable, TRUE);
1592         if (ACPI_FAILURE (Status))
1593         {
1594             return (Status);
1595         }

1597         DtInsertSubtable (ParentTable, Subtable);
1598         SubtableCount--;
1599     }

1601     DtPopSubtable ();
1602     return (AE_OK);
1603 }


1606 /******************************************************************************
1607  *
1608  * FUNCTION:    DtCompileMsct
1609  *
1610  * PARAMETERS:  List                - Current field list pointer
1611  *
1612  * RETURN:      Status
1613  *
1614  * DESCRIPTION: Compile MSCT.
1615  *
1616  *****************************************************************************/

1618 ACPI_STATUS
1619 DtCompileMsct (
1620     void                    **List)
1621 {
1622     ACPI_STATUS             Status;


1625     Status = DtCompileTwoSubtables (List,
1626                 AcpiDmTableInfoMsct, AcpiDmTableInfoMsct0);
1627     return (Status);
1628 }


1631 /******************************************************************************
1632  *
1633  * FUNCTION:    DtCompileMtmr
1634  *
1635  * PARAMETERS:  List                - Current field list pointer
1636  *
1637  * RETURN:      Status
1638  *
1639  * DESCRIPTION: Compile MTMR.
1640  *
1641  *****************************************************************************/

1643 ACPI_STATUS
```



```
1644 DtCompileMtmr (
1645     void                    **List)
1646 {
1647     ACPI_STATUS             Status;


1650     Status = DtCompileTwoSubtables (List,
1651                 AcpiDmTableInfoMtmr, AcpiDmTableInfoMtmr0);
1652     return (Status);
1653 }


1656 /******************************************************************************
1657  *
1658  * FUNCTION:    DtCompilePcct
1659  *
1660  * PARAMETERS:  List                - Current field list pointer
1661  *
1662  * RETURN:      Status
1663  *
1664  * DESCRIPTION: Compile PCCT.
1665  *
1666  *****************************************************************************/

1668 ACPI_STATUS
1669 DtCompilePcct (
1670     void                    **List)
1671 {
1672     ACPI_STATUS             Status;
1673     DT_SUBTABLE             *Subtable;
1674     DT_SUBTABLE             *ParentTable;
1675     DT_FIELD                **PFieldList = (DT_FIELD **) List;
1676     DT_FIELD                *SubtableStart;
1677     ACPI_SUBTABLE_HEADER    *PcctHeader;
1678     ACPI_DMTABLE_INFO       *InfoTable;


1681     Status = DtCompileTable (PFieldList, AcpiDmTableInfoPcct,
1682                 &Subtable, TRUE);
1683     if (ACPI_FAILURE (Status))
1684     {
1685         return (Status);
1686     }

1688     ParentTable = DtPeekSubtable ();
1689     DtInsertSubtable (ParentTable, Subtable);

1691     while (*PFieldList)
1692     {
1693         SubtableStart = *PFieldList;
1694         Status = DtCompileTable (PFieldList, AcpiDmTableInfoPcctHdr,
1695                     &Subtable, TRUE);
1696         if (ACPI_FAILURE (Status))
1697         {
1698             return (Status);
1699         }

1701         ParentTable = DtPeekSubtable ();
1702         DtInsertSubtable (ParentTable, Subtable);
1703         DtPushSubtable (Subtable);

1705         PcctHeader = ACPI_CAST_PTR (ACPI_SUBTABLE_HEADER, Subtable->Buffer);

1707         switch (PcctHeader->Type)
1708         {
1709         case ACPI_PCCT_TYPE_GENERIC_SUBSPACE:
```

```
            InfoTable = AcpiDmTableInfoPcct0;
            break;

        default:

            DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "PCCT");
            return (AE_ERROR);
        }

        Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPopSubtable ();
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompilePmtt
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile PMTT.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompilePmtt (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    DT_FIELD                *SubtableStart;
    ACPI_PMTT_HEADER        *PmttHeader;
    ACPI_PMTT_CONTROLLER    *PmttController;
    UINT16                  DomainCount;
    UINT8                   PrevType = ACPI_PMTT_TYPE_SOCKET;


    /* Main table */

    Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt, &Subtable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParentTable = DtPeekSubtable ();
    DtInsertSubtable (ParentTable, Subtable);
    DtPushSubtable (Subtable);

    while (*PFieldList)
    {
```



```
        SubtableStart = *PFieldList;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmttHdr,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        PmttHeader = ACPI_CAST_PTR (ACPI_PMTT_HEADER, Subtable->Buffer);
        while (PrevType >= PmttHeader->Type)
        {
            DtPopSubtable ();

            if (PrevType == ACPI_PMTT_TYPE_SOCKET)
            {
                break;
            }
            PrevType--;
        }
        PrevType = PmttHeader->Type;

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        switch (PmttHeader->Type)
        {
        case ACPI_PMTT_TYPE_SOCKET:

            /* Subtable: Socket Structure */

            Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt0,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            ParentTable = DtPeekSubtable ();
            DtInsertSubtable (ParentTable, Subtable);
            break;

        case ACPI_PMTT_TYPE_CONTROLLER:

            /* Subtable: Memory Controller Structure */

            Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt1,
                        &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            ParentTable = DtPeekSubtable ();
            DtInsertSubtable (ParentTable, Subtable);

            PmttController = ACPI_CAST_PTR (ACPI_PMTT_CONTROLLER,
                (Subtable->Buffer - sizeof (ACPI_PMTT_HEADER)));
            DomainCount = PmttController->DomainCount;

            while (DomainCount)
            {
                Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt1a,
                    &Subtable, TRUE);
                if (ACPI_FAILURE (Status))
                {
```

```
                    return (Status);
                }

                DtInsertSubtable (ParentTable, Subtable);
                DomainCount--;
            }
            break;

        case ACPI_PMTT_TYPE_DIMM:

            /* Subtable: Physical Component Structure */

            Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt2,
                    &Subtable, TRUE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            ParentTable = DtPeekSubtable ();
            DtInsertSubtable (ParentTable, Subtable);
            break;

        default:

            DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "PMTT");
            return (AE_ERROR);
        }
    }

    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileRsdt
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile RSDT.
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileRsdt (
    void                    **List)
{
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                *FieldList = *(DT_FIELD **) List;
    UINT32                  Address;


    ParentTable = DtPeekSubtable ();

    while (FieldList)
    {
        DtCompileInteger ((UINT8 *) &Address, FieldList, 4, DT_NON_ZERO);

        DtCreateSubtable ((UINT8 *) &Address, 4, &Subtable);
        DtInsertSubtable (ParentTable, Subtable);
        FieldList = FieldList->Next;
    }
```



```
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileS3pt
 *
 * PARAMETERS:  PFieldList          - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile S3PT (Pointed to by FPDT)
 *
 *****************************************************************************/

ACPI_STATUS
DtCompileS3pt (
    DT_FIELD                **PFieldList)
{
    ACPI_STATUS             Status;
    ACPI_S3PT_HEADER        *S3ptHeader;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    ACPI_DMTABLE_INFO       *InfoTable;
    DT_FIELD                *SubtableStart;


    Status = DtCompileTable (PFieldList, AcpiDmTableInfoS3pt,
                &Gbl_RootTable, TRUE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    DtPushSubtable (Gbl_RootTable);

    while (*PFieldList)
    {
        SubtableStart = *PFieldList;
        Status = DtCompileTable (PFieldList, AcpiDmTableInfoS3ptHdr,
                    &Subtable, TRUE);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        ParentTable = DtPeekSubtable ();
        DtInsertSubtable (ParentTable, Subtable);
        DtPushSubtable (Subtable);

        S3ptHeader = ACPI_CAST_PTR (ACPI_S3PT_HEADER, Subtable->Buffer);

        switch (S3ptHeader->Type)
        {
        case ACPI_S3PT_TYPE_RESUME:

            InfoTable = AcpiDmTableInfoS3pt0;
            break;

        case ACPI_S3PT_TYPE_SUSPEND:

            InfoTable = AcpiDmTableInfoS3pt1;
            break;
```

```
1974         default:

1976             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "S3PT");
1977             return (AE_ERROR);
1978         }

1980         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1981         if (ACPI_FAILURE (Status))
1982         {
1983             return (Status);
1984         }

1986         ParentTable = DtPeekSubtable ();
1987         DtInsertSubtable (ParentTable, Subtable);
1988         DtPopSubtable ();
1989     }

1991     return (AE_OK);
1992 }


1995 /******************************************************************************
1996  *
1997  * FUNCTION:    DtCompileSlic
1998  *
1999  * PARAMETERS:  List                - Current field list pointer
2000  *
2001  * RETURN:      Status
2002  *
2003  * DESCRIPTION: Compile SLIC.
2004  *
2005  *****************************************************************************/

2007 ACPI_STATUS
2008 DtCompileSlic (
2009     void                    **List)
2010 {
2011     ACPI_STATUS             Status;
2012     DT_SUBTABLE             *Subtable;
2013     DT_SUBTABLE             *ParentTable;
2014     DT_FIELD                **PFieldList = (DT_FIELD **) List;
2015     DT_FIELD                *SubtableStart;
2016     ACPI_SLIC_HEADER        *SlicHeader;
2017     ACPI_DMTABLE_INFO       *InfoTable;


2020     while (*PFieldList)
2021     {
2022         SubtableStart = *PFieldList;
2023         Status = DtCompileTable (PFieldList, AcpiDmTableInfoSlicHdr,
2024                     &Subtable, TRUE);
2025         if (ACPI_FAILURE (Status))
2026         {
2027             return (Status);
2028         }

2030         ParentTable = DtPeekSubtable ();
2031         DtInsertSubtable (ParentTable, Subtable);
2032         DtPushSubtable (Subtable);

2034         SlicHeader = ACPI_CAST_PTR (ACPI_SLIC_HEADER, Subtable->Buffer);

2036         switch (SlicHeader->Type)
2037         {
2038         case ACPI_SLIC_TYPE_PUBLIC_KEY:
```



```
2040             InfoTable = AcpiDmTableInfoSlic0;
2041             break;

2043         case ACPI_SLIC_TYPE_WINDOWS_MARKER:

2045             InfoTable = AcpiDmTableInfoSlic1;
2046             break;

2048         default:

2050             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "SLIC");
2051             return (AE_ERROR);
2052         }

2054         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
2055         if (ACPI_FAILURE (Status))
2056         {
2057             return (Status);
2058         }

2060         ParentTable = DtPeekSubtable ();
2061         DtInsertSubtable (ParentTable, Subtable);
2062         DtPopSubtable ();
2063     }

2065     return (AE_OK);
2066 }


2069 /******************************************************************************
2070  *
2071  * FUNCTION:    DtCompileSlit
2072  *
2073  * PARAMETERS:  List                - Current field list pointer
2074  *
2075  * RETURN:      Status
2076  *
2077  * DESCRIPTION: Compile SLIT.
2078  *
2079  *****************************************************************************/

2081 ACPI_STATUS
2082 DtCompileSlit (
2083     void                    **List)
2084 {
2085     ACPI_STATUS             Status;
2086     DT_SUBTABLE             *Subtable;
2087     DT_SUBTABLE             *ParentTable;
2088     DT_FIELD                **PFieldList = (DT_FIELD **) List;
2089     DT_FIELD                *FieldList;
2090     UINT32                  Localities;
2091     UINT8                   *LocalityBuffer;


2094     Status = DtCompileTable (PFieldList, AcpiDmTableInfoSlit,
2095                 &Subtable, TRUE);
2096     if (ACPI_FAILURE (Status))
2097     {
2098         return (Status);
2099     }

2101     ParentTable = DtPeekSubtable ();
2102     DtInsertSubtable (ParentTable, Subtable);

2104     Localities = *ACPI_CAST_PTR (UINT32, Subtable->Buffer);
2105     LocalityBuffer = UtLocalCalloc (Localities);
```

```
2107     /* Compile each locality buffer */

2109     FieldList = *PFieldList;
2110     while (FieldList)
2111     {
2112         DtCompileBuffer (LocalityBuffer,
2113             FieldList->Value, FieldList, Localities);

2115         DtCreateSubtable (LocalityBuffer, Localities, &Subtable);
2116         DtInsertSubtable (ParentTable, Subtable);
2117         FieldList = FieldList->Next;
2118     }

2120     ACPI_FREE (LocalityBuffer);
2121     return (AE_OK);
2122 }


2125 /******************************************************************************
2126  *
2127  * FUNCTION:    DtCompileSrat
2128  *
2129  * PARAMETERS:  List                - Current field list pointer
2130  *
2131  * RETURN:      Status
2132  *
2133  * DESCRIPTION: Compile SRAT.
2134  *
2135  *****************************************************************************/

2137 ACPI_STATUS
2138 DtCompileSrat (
2139     void                    **List)
2140 {
2141     ACPI_STATUS             Status;
2142     DT_SUBTABLE             *Subtable;
2143     DT_SUBTABLE             *ParentTable;
2144     DT_FIELD                **PFieldList = (DT_FIELD **) List;
2145     DT_FIELD                *SubtableStart;
2146     ACPI_SUBTABLE_HEADER    *SratHeader;
2147     ACPI_DMTABLE_INFO       *InfoTable;


2150     Status = DtCompileTable (PFieldList, AcpiDmTableInfoSrat,
2151                 &Subtable, TRUE);
2152     if (ACPI_FAILURE (Status))
2153     {
2154         return (Status);
2155     }

2157     ParentTable = DtPeekSubtable ();
2158     DtInsertSubtable (ParentTable, Subtable);

2160     while (*PFieldList)
2161     {
2162         SubtableStart = *PFieldList;
2163         Status = DtCompileTable (PFieldList, AcpiDmTableInfoSratHdr,
2164                     &Subtable, TRUE);
2165         if (ACPI_FAILURE (Status))
2166         {
2167             return (Status);
2168         }

2170         ParentTable = DtPeekSubtable ();
2171         DtInsertSubtable (ParentTable, Subtable);
```



```
2172         DtPushSubtable (Subtable);

2174         SratHeader = ACPI_CAST_PTR (ACPI_SUBTABLE_HEADER, Subtable->Buffer);

2176         switch (SratHeader->Type)
2177         {
2178         case ACPI_SRAT_TYPE_CPU_AFFINITY:

2180             InfoTable = AcpiDmTableInfoSrat0;
2181             break;

2183         case ACPI_SRAT_TYPE_MEMORY_AFFINITY:

2185             InfoTable = AcpiDmTableInfoSrat1;
2186             break;

2188         case ACPI_SRAT_TYPE_X2APIC_CPU_AFFINITY:

2190             InfoTable = AcpiDmTableInfoSrat2;
2191             break;

2193         default:

2195             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "SRAT");
2196             return (AE_ERROR);
2197         }

2199         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
2200         if (ACPI_FAILURE (Status))
2201         {
2202             return (Status);
2203         }

2205         ParentTable = DtPeekSubtable ();
2206         DtInsertSubtable (ParentTable, Subtable);
2207         DtPopSubtable ();
2208     }

2210     return (AE_OK);
2211 }


2214 /******************************************************************************
2215  *
2216  * FUNCTION:    DtGetGenericTableInfo
2217  *
2218  * PARAMETERS:  Name                - Generic type name
2219  *
2220  * RETURN:      Info entry
2221  *
2222  * DESCRIPTION: Obtain table info for a generic name entry
2223  *
2224  *****************************************************************************/

2226 ACPI_DMTABLE_INFO *
2227 DtGetGenericTableInfo (
2228     char                    *Name)
2229 {
2230     ACPI_DMTABLE_INFO       *Info;
2231     UINT32                  i;


2234     if (!Name)
2235     {
2236         return (NULL);
2237     }
```

```
2239     /* Search info table for name match */

2241     for (i = 0; ; i++)
2242     {
2243         Info = AcpiDmTableInfoGeneric[i];
2244         if (Info->Opcode == ACPI_DMT_EXIT)
2245         {
2246             Info = NULL;
2247             break;
2248         }

2250         /* Use caseless compare for generic keywords */

2252         if (!AcpiUtStricmp (Name, Info->Name))
2253         {
2254             break;
2255         }
2256     }

2258     return (Info);
2259 }


2262 /******************************************************************************
2263  *
2264  * FUNCTION:    DtCompileUefi
2265  *
2266  * PARAMETERS:  List                - Current field list pointer
2267  *
2268  * RETURN:      Status
2269  *
2270  * DESCRIPTION: Compile UEFI.
2271  *
2272  *****************************************************************************/

2274 ACPI_STATUS
2275 DtCompileUefi (
2276     void                    **List)
2277 {
2278     ACPI_STATUS             Status;
2279     DT_SUBTABLE             *Subtable;
2280     DT_SUBTABLE             *ParentTable;
2281     DT_FIELD                **PFieldList = (DT_FIELD **) List;
2282     UINT16                  *DataOffset;


2285     /* Compile the predefined portion of the UEFI table */

2287     Status = DtCompileTable (PFieldList, AcpiDmTableInfoUefi,
2288                 &Subtable, TRUE);
2289     if (ACPI_FAILURE (Status))
2290     {
2291         return (Status);
2292     }

2294     DataOffset = (UINT16 *) (Subtable->Buffer + 16);
2295     *DataOffset = sizeof (ACPI_TABLE_UEFI);

2297     ParentTable = DtPeekSubtable ();
2298     DtInsertSubtable (ParentTable, Subtable);

2300     /*
2301      * Compile the "generic" portion of the UEFI table. This
2302      * part of the table is not predefined and any of the generic
2303      * operators may be used.
```



```
2304      */

2306     DtCompileGeneric ((void **) PFieldList);

2308     return (AE_OK);
2309 }


2312 /******************************************************************************
2313  *
2314  * FUNCTION:    DtCompileVrtc
2315  *
2316  * PARAMETERS:  List                - Current field list pointer
2317  *
2318  * RETURN:      Status
2319  *
2320  * DESCRIPTION: Compile VRTC.
2321  *
2322  *****************************************************************************/

2324 ACPI_STATUS
2325 DtCompileVrtc (
2326     void                    **List)
2327 {
2328     ACPI_STATUS             Status;


2331     Status = DtCompileTwoSubtables (List,
2332                 AcpiDmTableInfoVrtc, AcpiDmTableInfoVrtc0);
2333     return (Status);
2334 }


2337 /******************************************************************************
2338  *
2339  * FUNCTION:    DtCompileWdat
2340  *
2341  * PARAMETERS:  List                - Current field list pointer
2342  *
2343  * RETURN:      Status
2344  *
2345  * DESCRIPTION: Compile WDAT.
2346  *
2347  *****************************************************************************/

2349 ACPI_STATUS
2350 DtCompileWdat (
2351     void                    **List)
2352 {
2353     ACPI_STATUS             Status;


2356     Status = DtCompileTwoSubtables (List,
2357                 AcpiDmTableInfoWdat, AcpiDmTableInfoWdat0);
2358     return (Status);
2359 }


2362 /******************************************************************************
2363  *
2364  * FUNCTION:    DtCompileXsdt
2365  *
2366  * PARAMETERS:  List                - Current field list pointer
2367  *
2368  * RETURN:      Status
2369  *
```

```
 * DESCRIPTION: Compile XSDT.
 *
 ******************************************************************************/

ACPI_STATUS
DtCompileXsdt (
    void                    **List)
{
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                *FieldList = *(DT_FIELD **) List;
    UINT64                  Address;

    ParentTable = DtPeekSubtable ();

    while (FieldList)
    {
        DtCompileInteger ((UINT8 *) &Address, FieldList, 8, DT_NON_ZERO);

        DtCreateSubtable ((UINT8 *) &Address, 8, &Subtable);
        DtInsertSubtable (ParentTable, Subtable);
        FieldList = FieldList->Next;
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    DtCompileGeneric
 *
 * PARAMETERS:  List                - Current field list pointer
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Compile generic unknown table.
 *
 ******************************************************************************/

ACPI_STATUS
DtCompileGeneric (
    void                    **List)
{
    ACPI_STATUS             Status;
    DT_SUBTABLE             *Subtable;
    DT_SUBTABLE             *ParentTable;
    DT_FIELD                **PFieldList = (DT_FIELD **) List;
    ACPI_DMTABLE_INFO       *Info;


    ParentTable = DtPeekSubtable ();

    /*
     * Compile the "generic" portion of the table. This
     * part of the table is not predefined and any of the generic
     * operators may be used.
     */

    /* Find any and all labels in the entire generic portion */

    DtDetectAllLabels (*PFieldList);

    /* Now we can actually compile the parse tree */

    while (*PFieldList)
```



```
    {
        Info = DtGetGenericTableInfo ((*PFieldList)->Name);
        if (!Info)
        {
            sprintf (MsgBuffer, "Generic data type \"%s\" not found",
                (*PFieldList)->Name);
            DtNameError (ASL_ERROR, ASL_MSG_INVALID_FIELD_NAME,
                (*PFieldList), MsgBuffer);

            *PFieldList = (*PFieldList)->Next;
            continue;
        }

        Status = DtCompileTable (PFieldList, Info,
                    &Subtable, TRUE);
        if (ACPI_SUCCESS (Status))
        {
            DtInsertSubtable (ParentTable, Subtable);
        }
        else
        {
            *PFieldList = (*PFieldList)->Next;

            if (Status == AE_NOT_FOUND)
            {
                sprintf (MsgBuffer, "Generic data type \"%s\" not found",
                    (*PFieldList)->Name);
                DtNameError (ASL_ERROR, ASL_MSG_INVALID_FIELD_NAME,
                    (*PFieldList), MsgBuffer);
            }
        }
    }

    return (AE_OK);
}
```

```
**********************************************************
   11365 Thu Dec 26 13:48:38 2013
new/usr/src/common/acpica/compiler/dttemplate.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dttemplate - ACPI table template generation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "acapps.h"
#include "dtcompiler.h"
#include "dttemplate.h" /* Contains the hex ACPI table templates */

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dttemplate")


/* Local prototypes */

static BOOLEAN
AcpiUtIsSpecialTable (
    char                    *Signature);

static ACPI_STATUS
DtCreateOneTemplate (
    char                    *Signature,
    ACPI_DMTABLE_DATA       *TableData);

static ACPI_STATUS
DtCreateAllTemplates (
    void);


/******************************************************************************
 *
 * FUNCTION:    AcpiUtIsSpecialTable
 *
 * PARAMETERS:  Signature           - ACPI table signature
 *
 * RETURN:      TRUE if signature is a special ACPI table
 *
 * DESCRIPTION: Check for valid ACPI tables that are not in the main ACPI
 *              table data structure (AcpiDmTableData).
 *
 *****************************************************************************/

static BOOLEAN
AcpiUtIsSpecialTable (
    char                    *Signature)
{

    if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT) ||
        ACPI_COMPARE_NAME (Signature, ACPI_SIG_SSDT) ||
        ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS) ||
        ACPI_COMPARE_NAME (Signature, ACPI_RSDP_NAME))
    {
        return (TRUE);
    }

    return (FALSE);
}


/******************************************************************************
 *
 * FUNCTION:    DtCreateTemplates
 *
 * PARAMETERS:  Signature           - ACPI table signature
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create one or more template files.
 *
 *****************************************************************************/

ACPI_STATUS
DtCreateTemplates (
    char                    *Signature)
{
    ACPI_DMTABLE_DATA       *TableData;
    ACPI_STATUS             Status;


    AslInitializeGlobals ();

    /* Default (no signature) is DSDT */

    if (!Signature)
    {
        Signature = "DSDT";
        goto GetTemplate;
```

```
127     }

129     AcpiUtStrupr (Signature);
130     if (!ACPI_STRCMP (Signature, "ALL") ||
131         !ACPI_STRCMP (Signature, "*"))
132     {
133         /* Create all available/known templates */

135         Status = DtCreateAllTemplates ();
136         return (Status);
137     }

139     /*
140      * Validate signature and get the template data:
141      *  1) Signature must be 4 characters
142      *  2) Signature must be a recognized ACPI table
143      *  3) There must be a template associated with the signature
144      */
145     if (strlen (Signature) != ACPI_NAME_SIZE)
146     {
147         fprintf (stderr,
148             "%s: Invalid ACPI table signature (length must be 4 characters)\n",
149             Signature);
150         return (AE_ERROR);
151     }

153     /*
154      * Some slack for the two strange tables whose name is different than
155      * their signatures: MADT->APIC and FADT->FACP.
156      */
157     if (!strcmp (Signature, "MADT"))
158     {
159         Signature = "APIC";
160     }
161     else if (!strcmp (Signature, "FADT"))
162     {
163         Signature = "FACP";
164     }

166 GetTemplate:
167     TableData = AcpiDmGetTableData (Signature);
168     if (TableData)
169     {
170         if (!TableData->Template)
171         {
172             fprintf (stderr, "%4.4s: No template available\n", Signature);
173             return (AE_ERROR);
174         }
175     }
176     else if (!AcpiUtIsSpecialTable (Signature))
177     {
178         fprintf (stderr,
179             "%4.4s: Unrecognized ACPI table signature\n", Signature);
180         return (AE_ERROR);
181     }

183     Status = AdInitialize ();
184     if (ACPI_FAILURE (Status))
185     {
186         return (Status);
187     }

189     Status = DtCreateOneTemplate (Signature, TableData);
190     return (Status);
191 }
```



```
194 /******************************************************************************
195  *
196  * FUNCTION:    DtCreateAllTemplates
197  *
198  * PARAMETERS:  None
199  *
200  * RETURN:      Status
201  *
202  * DESCRIPTION: Create all currently defined template files
203  *
204  *****************************************************************************/

206 static ACPI_STATUS
207 DtCreateAllTemplates (
208     void)
209 {
210     ACPI_DMTABLE_DATA       *TableData;
211     ACPI_STATUS             Status;


214     Status = AdInitialize ();
215     if (ACPI_FAILURE (Status))
216     {
217         return (Status);
218     }

220     fprintf (stderr, "Creating all supported Template files\n");

222     /* Walk entire ACPI table data structure */

224     for (TableData = AcpiDmTableData; TableData->Signature; TableData++)
225     {
226         /* If table has a template, create the template file */

228         if (TableData->Template)
229         {
230             Status = DtCreateOneTemplate (TableData->Signature,
231                         TableData);
232             if (ACPI_FAILURE (Status))
233             {
234                 return (Status);
235             }
236         }
237     }

239     /*
240      * Create the special ACPI tables:
241      * 1) DSDT/SSDT are AML tables, not data tables
242      * 2) FACS and RSDP have non-standard headers
243      */
244     Status = DtCreateOneTemplate (ACPI_SIG_DSDT, NULL);
245     if (ACPI_FAILURE (Status))
246     {
247         return (Status);
248     }

250     Status = DtCreateOneTemplate (ACPI_SIG_SSDT, NULL);
251     if (ACPI_FAILURE (Status))
252     {
253         return (Status);
254     }

256     Status = DtCreateOneTemplate (ACPI_SIG_FACS, NULL);
257     if (ACPI_FAILURE (Status))
258     {
```

```
259         return (Status);
260     }

262     Status = DtCreateOneTemplate (ACPI_RSDP_NAME, NULL);
263     if (ACPI_FAILURE (Status))
264     {
265         return (Status);
266     }

268     return (AE_OK);
269 }


272 /******************************************************************************
273  *
274  * FUNCTION:    DtCreateOneTemplate
275  *
276  * PARAMETERS:  Signature           - ACPI signature, NULL terminated.
277  *              TableData           - Entry in ACPI table data structure.
278  *                                    NULL if a special ACPI table.
279  *
280  * RETURN:      Status
281  *
282  * DESCRIPTION: Create one template source file for the requested ACPI table.
283  *
284  *****************************************************************************/

286 static ACPI_STATUS
287 DtCreateOneTemplate (
288     char                    *Signature,
289     ACPI_DMTABLE_DATA       *TableData)
290 {
291     char                    *DisasmFilename;
292     FILE                    *File;
293     ACPI_STATUS             Status = AE_OK;
294     ACPI_SIZE               Actual;


297     /* New file will have a .asl suffix */

299     DisasmFilename = FlGenerateFilename (
300         Signature, FILE_SUFFIX_ASL_CODE);
301     if (!DisasmFilename)
302     {
303         fprintf (stderr, "Could not generate output filename\n");
304         return (AE_ERROR);
305     }

307     /* Probably should prompt to overwrite the file */

309     AcpiUtStrlwr (DisasmFilename);
310     File = fopen (DisasmFilename, "w+");
311     if (!File)
312     {
313         fprintf (stderr, "Could not open output file %s\n", DisasmFilename);
314         return (AE_ERROR);
315     }

317     /* Emit the common file header */

319     AcpiOsRedirectOutput (File);

321     AcpiOsPrintf ("/*\n");
322     AcpiOsPrintf (ACPI_COMMON_HEADER ("iASL Compiler/Disassembler", " * "));

324     AcpiOsPrintf (" * Template for [%4.4s] ACPI Table\n",
```



```
325         Signature);

327     /* Dump the actual ACPI table */

329     if (TableData)
330     {
331         /* Normal case, tables that appear in AcpiDmTableData */

333         if (Gbl_VerboseTemplates)
334         {
335             AcpiOsPrintf (" * Format: [HexOffset DecimalOffset ByteLength]"
336                 "  FieldName : HexFieldValue\n */\n\n");
337         }
338         else
339         {
340             AcpiOsPrintf (" * Format: [ByteLength]"
341                 "  FieldName : HexFieldValue\n */\n\n");
342         }

344         AcpiDmDumpDataTable (ACPI_CAST_PTR (ACPI_TABLE_HEADER,
345             TableData->Template));
346     }
347     else
348     {
349         /* Special ACPI tables - DSDT, SSDT, FADT, RSDP */

351         AcpiOsPrintf (" */\n\n");
352         if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT))
353         {
354             Actual = fwrite (TemplateDsdt, 1, sizeof (TemplateDsdt) -1, File);
355             if (Actual != sizeof (TemplateDsdt) -1)
356             {
357                 fprintf (stderr,
358                     "Could not write to output file %s\n", DisasmFilename);
359                 Status = AE_ERROR;
360                 goto Cleanup;
361             }
362         }
363         else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_SSDT))
364         {
365             Actual = fwrite (TemplateSsdt, 1, sizeof (TemplateSsdt) -1, File);
366             if (Actual != sizeof (TemplateSsdt) -1)
367             {
368                 fprintf (stderr,
369                     "Could not write to output file %s\n", DisasmFilename);
370                 Status = AE_ERROR;
371                 goto Cleanup;
372             }
373         }
374         else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS)) /* FADT */
375         {
376             AcpiDmDumpDataTable (ACPI_CAST_PTR (ACPI_TABLE_HEADER,
377                 TemplateFacs));
378         }
379         else if (ACPI_COMPARE_NAME (Signature, ACPI_RSDP_NAME))
380         {
381             AcpiDmDumpDataTable (ACPI_CAST_PTR (ACPI_TABLE_HEADER,
382                 TemplateRsdp));
383         }
384         else
385         {
386             fprintf (stderr,
387                 "%4.4s, Unrecognized ACPI table signature\n", Signature);
388             Status = AE_ERROR;
389             goto Cleanup;
390         }
```

```
391     }

393     fprintf (stderr,
394         "Created ACPI table template for [%4.4s], written to \"%s\"\n",
395         Signature, DisasmFilename);

397 Cleanup:
398     fclose (File);
399     AcpiOsRedirectOutput (stdout);
400     ACPI_FREE (DisasmFilename);
401     return (Status);
402 }
```

```
**********************************************************
   63848 Thu Dec 26 13:48:38 2013
new/usr/src/common/acpica/compiler/dttemplate.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dttemplate.h - ACPI table template definitions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __DTTEMPLATE_H
#define __DTTEMPLATE_H


/* Special templates for DSDT and SSDT (AML byte-code tables) */

const char TemplateDsdt[] =
    "DefinitionBlock (\"dsdt.aml\", \"DSDT\", 2, \"Intel\", \"Template\", 0x0000
    "{\n"
    "    Method (MAIN, 0, NotSerialized)\n"
    "    {\n"
    "        Return (Zero)\n"
    "    }\n"
    "}\n\n";

const char TemplateSsdt[] =
```



```
    "DefinitionBlock (\"ssdt.aml\", \"SSDT\", 2, \"Intel\", \"Template\", 0x0000
    "{\n"
    "    Method (MAIN, 0, NotSerialized)\n"
    "    {\n"
    "        Return (Zero)\n"
    "    }\n"
    "}\n\n";


/* Templates for ACPI data tables */

const unsigned char TemplateAsf[] =
{
    0x41,0x53,0x46,0x21,0x72,0x00,0x00,0x00,  /* 00000000    "ASF!r..." */
    0x10,0x0B,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x10,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x14,0x00,  /* 00000030    "........" */
    0x00,0x00,0x01,0x0C,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x02,0x00,0x0C,0x00,0x01,0x04,0x00,0x00,  /* 00000048    "........" */
    0x00,0x00,0x00,0x00,0x03,0x00,0x17,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x84,0x00,0x07,0x00,0x00,  /* 00000068    "........" */
    0x01,0x00                                 /* 00000070    ".."       */
};

const unsigned char TemplateBgrt[] =
{
    0x42,0x47,0x52,0x54,0x38,0x00,0x00,0x00,  /* 00000000    "BGRT8..." */
    0x01,0x0D,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x23,0x06,0x11,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "#.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000030    "........" */
};

const unsigned char TemplateBert[] =
{
    0x42,0x45,0x52,0x54,0x30,0x00,0x00,0x00,  /* 00000000    "BERT0..." */
    0x01,0x15,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000028    "........" */
};

const unsigned char TemplateBoot[] =
{
    0x42,0x4F,0x4F,0x54,0x28,0x00,0x00,0x00,  /* 00000000    "BOOT(..." */
    0x01,0x0D,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x04,0x06,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00   /* 00000020    "(.. ...." */
};

const unsigned char TemplateCpep[] =
{
    0x43,0x50,0x45,0x50,0x34,0x00,0x00,0x00,  /* 00000000    "CPEP4..." */
    0x01,0x0F,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
```

```
126     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
127     0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,  /* 00000028    "........" */
128     0x00,0x00,0x00,0x00                       /* 00000030    "...."     */
129 };

131 const unsigned char TemplateCsrt[] =
132 {
133     0x43,0x53,0x52,0x54,0x4C,0x01,0x00,0x00,  /* 00000000    "CSRTL..." */
134     0x01,0x0D,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
135     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
136     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
137     0x14,0x11,0x12,0x20,0x88,0x00,0x00,0x00,  /* 00000020    "... ...." */
138     0x49,0x4E,0x54,0x4C,0x00,0x00,0x00,0x00,  /* 00000028    "INTL...." */
139     0x60,0x9C,0x00,0x00,0x02,0x00,0x00,0x00,  /* 00000030    "`......." */
140     0x1C,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000038    "........" */
141     0x00,0x00,0xA0,0xB3,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
142     0x2A,0x00,0x00,0x00,0x02,0x00,0x06,0x20,  /* 00000048    "*...... " */
143     0x00,0x00,0x10,0x00,0xFF,0x0F,0x00,0x00,  /* 00000050    "........" */
144     0x0C,0x00,0x00,0x00,0x03,0x00,0x01,0x00,  /* 00000058    "........" */
145     0x53,0x50,0x49,0x20,0x0C,0x00,0x00,0x00,  /* 00000060    "SPI ...." */
146     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x30,  /* 00000068    "....CHA0" */
147     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 00000070    "........" */
148     0x43,0x48,0x41,0x31,0x0C,0x00,0x00,0x00,  /* 00000078    "CHA1...." */
149     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x32,  /* 00000080    "....CHA2" */
150     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 00000088    "........" */
151     0x43,0x48,0x41,0x33,0x0C,0x00,0x00,0x00,  /* 00000090    "CHA3...." */
152     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x34,  /* 00000098    "....CHA4" */
153     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 000000A0    "........" */
154     0x43,0x48,0x41,0x35,0xA0,0x00,0x00,0x00,  /* 000000A8    "CHA5...." */
155     0x49,0x4E,0x54,0x4C,0x00,0x00,0x00,0x00,  /* 000000B0    "INTL...." */
156     0x60,0x9C,0x00,0x00,0x03,0x00,0x00,0x00,  /* 000000B8    "`......." */
157     0x1C,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 000000C0    "........" */
158     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
159     0x2B,0x00,0x00,0x00,0x02,0x00,0x08,0x20,  /* 000000D0    "+...... " */
160     0x10,0x00,0x10,0x00,0xFF,0x0F,0x00,0x00,  /* 000000D8    "........" */
161     0x0C,0x00,0x00,0x00,0x03,0x00,0x01,0x00,  /* 000000E0    "........" */
162     0x49,0x32,0x43,0x20,0x0C,0x00,0x00,0x00,  /* 000000E8    "I2C ...." */
163     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x30,  /* 000000F0    "....CHA0" */
164     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 000000F8    "........" */
165     0x43,0x48,0x41,0x31,0x0C,0x00,0x00,0x00,  /* 00000100    "CHA1...." */
166     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x32,  /* 00000108    "....CHA2" */
167     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 00000110    "........" */
168     0x43,0x48,0x41,0x33,0x0C,0x00,0x00,0x00,  /* 00000118    "CHA3...." */
169     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x34,  /* 00000120    "....CHA4" */
170     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 00000128    "........" */
171     0x43,0x48,0x41,0x35,0x0C,0x00,0x00,0x00,  /* 00000130    "CHA5...." */
172     0x03,0x00,0x00,0x00,0x43,0x48,0x41,0x36,  /* 00000138    "....CHA6" */
173     0x0C,0x00,0x00,0x00,0x03,0x00,0x00,0x00,  /* 00000140    "........" */
174     0x43,0x48,0x41,0x37                       /* 00000148    "CHA7"     */
175 };

177 const unsigned char TemplateDbg2[] =
178 {
179     0x44,0x42,0x47,0x32,0xB2,0x00,0x00,0x00,  /* 00000000    "DBG2...." */
180     0x01,0xBA,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
181     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
182     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
183     0x15,0x11,0x13,0x20,0x2C,0x00,0x00,0x00,  /* 00000020    "... ,..." */
184     0x02,0x00,0x00,0x00,0xEE,0x3F,0x00,0x02,  /* 00000028    ".....?.." */
185     0x09,0x00,0x36,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "..6....." */
186     0x00,0x80,0x00,0x00,0x00,0x00,0x16,0x00,  /* 00000038    "........" */
187     0x2E,0x00,0x01,0x32,0x00,0x03,0x88,0x77,  /* 00000040    "...2...w" */
188     0x66,0x55,0x44,0x33,0x22,0x11,0x01,0x64,  /* 00000048    "fUD3"..d" */
189     0x00,0x04,0x11,0x00,0xFF,0xEE,0xDD,0xCC,  /* 00000050    "........" */
190     0xBB,0xAA,0x10,0x32,0x54,0x76,0x98,0xBA,  /* 00000058    "...2Tv.." */
191     0xDC,0xFE,0x4D,0x79,0x44,0x65,0x76,0x69,  /* 00000060    "..MyDevi" */
```



```
192     0x63,0x65,0x00,0xEE,0x47,0x00,0x01,0x11,  /* 00000068    "ce..G..." */
193     0x00,0x26,0x00,0x10,0x00,0x37,0x00,0x00,  /* 00000070    ".&...7.." */
194     0x80,0x00,0x00,0x00,0x00,0x16,0x00,0x22,  /* 00000078    "......."" */
195     0x00,0x01,0x64,0x00,0x04,0x11,0x00,0xFF,  /* 00000080    "..d....." */
196     0xEE,0xDD,0xCC,0xBB,0xAA,0x98,0xBA,0xDC,  /* 00000088    "........" */
197     0xFE,0x5C,0x5C,0x5F,0x53,0x42,0x5F,0x2E,  /* 00000090    ".\\_SB_." */
198     0x50,0x43,0x49,0x30,0x2E,0x44,0x42,0x47,  /* 00000098    "PCI0.DBG" */
199     0x50,0x00,0x41,0x42,0x43,0x44,0x45,0x46,  /* 000000A0    "P.ABCDEF" */
200     0x47,0x48,0x49,0x50,0x51,0x52,0x53,0x54,  /* 000000A8    "GHIPQRST" */
201     0x55,0x56                                 /* 000000B0    "UV"       */
202 };

204 const unsigned char TemplateDbgp[] =
205 {
206     0x44,0x42,0x47,0x50,0x34,0x00,0x00,0x00,  /* 00000000    "DBGP4..." */
207     0x01,0x1A,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
208     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
209     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
210     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
211     0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
212     0x00,0x00,0x00,0x00                       /* 00000030    "...."     */
213 };

215 const unsigned char TemplateDmar[] =
216 {
217     0x44,0x4D,0x41,0x52,0x8C,0x00,0x00,0x00,  /* 00000000    "DMAR...." */
218     0x01,0x03,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
219     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
220     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
221     0x17,0x05,0x13,0x20,0x2F,0x01,0x00,0x00,  /* 00000020    "... /..." */
222     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
223     0x00,0x00,0x18,0x00,0x01,0x00,0x00,0x00,  /* 00000030    "........" */
224     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
225     0x03,0x08,0x00,0x00,0x08,0x00,0x00,0x01,  /* 00000040    "........" */
226     0x01,0x00,0x20,0x00,0x00,0x00,0x00,0x00,  /* 00000048    ".. ....." */
227     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
228     0xFF,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
229     0x01,0x08,0x00,0x00,0x00,0x00,0x00,0x02,  /* 00000060    "........" */
230     0x02,0x00,0x10,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
231     0x02,0x08,0x00,0x00,0x00,0x00,0x00,0x03,  /* 00000070    "........" */
232     0x03,0x00,0x14,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
233     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
234     0x00,0x00,0x00,0x00                       /* 00000088    "...."     */
235 };

237 const unsigned char TemplateEcdt[] =
238 {
239     0x45,0x43,0x44,0x54,0x42,0x00,0x00,0x00,  /* 00000000    "ECDTB..." */
240     0x01,0x2D,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".-INTEL " */
241     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
242     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
243     0x28,0x05,0x10,0x20,0x01,0x08,0x00,0x00,  /* 00000020    "(.. ...." */
244     0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "f......." */
245     0x01,0x08,0x00,0x00,0x62,0x00,0x00,0x00,  /* 00000030    "....b..." */
246     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
247     0x09,0x00                                 /* 00000040    ".."       */
248 };

250 const unsigned char TemplateEinj[] =
251 {
252     0x45,0x49,0x4E,0x4A,0x30,0x01,0x00,0x00,  /* 00000000    "EINJ0..." */
253     0x01,0x09,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
254     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
255     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
256     0x28,0x05,0x10,0x20,0x30,0x00,0x00,0x00,  /* 00000020    "(.. 0..." */
257     0x00,0x00,0x00,0x00,0x0A,0x00,0x00,0x00,  /* 00000028    "........" */
```

```
    0x00,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000030    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000048    "........" */
    0x01,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000050    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000068    "........" */
    0x02,0x02,0x01,0x00,0x00,0x40,0x00,0x04,  /* 00000070    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000088    "........" */
    0x03,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000090    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000A8    "........" */
    0x04,0x03,0x01,0x00,0x00,0x40,0x00,0x04,  /* 000000B0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000C8    "........" */
    0x05,0x03,0x01,0x00,0x01,0x10,0x00,0x02,  /* 000000D0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000D8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000E8    "........" */
    0x06,0x01,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000000F0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000108    "........" */
    0x07,0x00,0x01,0x00,0x00,0x40,0x00,0x04,  /* 00000110    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000118    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000120    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF   /* 00000128    "........" */
};

const unsigned char TemplateErst[] =
{
    0x45,0x52,0x53,0x54,0x30,0x02,0x00,0x00,  /* 00000000    "ERST0..." */
    0x01,0xAB,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x30,0x00,0x00,0x00,  /* 00000020    "(.. 0..." */
    0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x03,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000030    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000048    "........" */
    0x01,0x03,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000050    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000068    "........" */
    0x02,0x03,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000070    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000088    "........" */
    0x03,0x04,0x01,0x00,0x00,0x40,0x00,0x04,  /* 00000090    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000A8    "........" */
    0x04,0x02,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000000B0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000C8    "........" */
    0x05,0x03,0x00,0x00,0x01,0x08,0x00,0x01,  /* 000000D0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000D8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000000E8    "........" */
```



```
    0x06,0x01,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000000F0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000108    "........" */
    0x07,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000110    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000118    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000120    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000128    "........" */
    0x08,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000130    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000138    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000140    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000148    "........" */
    0x09,0x02,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000150    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000158    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000160    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000168    "........" */
    0x0A,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000170    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000178    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000180    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000188    "........" */
    0x0B,0x03,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000190    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000198    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001A0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000001A8    "........" */
    0x0C,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000001B0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001B8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001C0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000001C8    "........" */
    0x0D,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000001D0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001D8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001E0    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 000001E8    "........" */
    0x0E,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 000001F0    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001F8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000200    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,  /* 00000208    "........" */
    0x0F,0x00,0x00,0x00,0x00,0x40,0x00,0x04,  /* 00000210    ".....@.." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000218    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000220    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF   /* 00000228    "........" */
};

const unsigned char TemplateFacs[] =
{
    0x46,0x41,0x43,0x53,0x40,0x00,0x00,0x00,  /* 00000000    "FACS@..." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000008    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000010    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000018    "........" */
    0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000020    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000038    "........" */
};

/* Version 5 FADT */

const unsigned char TemplateFadt[] =
{
    0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00,  /* 00000000    "FACP...." */
    0x05,0x64,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".dINTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
```

```
390     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
391     0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000048    "........" */
392     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
393     0x04,0x02,0x01,0x04,0x08,0x00,0x00,0x00,  /* 00000058    "........" */
394     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
395     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
396     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01,  /* 00000070    "........" */
397     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
398     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000080    "........" */
399     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000088    "........" */
400     0x00,0x00,0x00,0x00,0x01,0x20,0x00,0x02,  /* 00000090    "..... .." */
401     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
402     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
403     0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x02,  /* 000000A8    "........" */
404     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B0    "........" */
405     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
406     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x00,  /* 000000C0    "........" */
407     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
408     0x01,0x20,0x00,0x03,0x01,0x00,0x00,0x00,  /* 000000D0    ". ......" */
409     0x00,0x00,0x00,0x00,0x01,0x40,0x00,0x01,  /* 000000D8    ".....@.." */
410     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
411     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
412     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01,  /* 000000F0    "........" */
413     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
414     0x01,0x08,0x00,0x01,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
415     0x00,0x00,0x00,0x00                       /* 00000108    "...."     */
416 };

418 const unsigned char TemplateFpdt[] =
419 {
420     0x46,0x50,0x44,0x54,0x64,0x00,0x00,0x00,  /* 00000000    "FPDTd..." */
421     0x01,0xBD,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
422     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
423     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
424     0x04,0x08,0x11,0x20,0x00,0x00,0x30,0x01,  /* 00000020    "... ..0." */
425     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
426     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
427     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
428     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
429     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
430     0x00,0x00,0x00,0x00,0x01,0x00,0x10,0x01,  /* 00000050    "........" */
431     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
432     0x00,0x00,0x00,0x00                       /* 00000060    "...."     */
433 };

435 const unsigned char TemplateGtdt[] =
436 {
437     0x47,0x54,0x44,0x54,0x50,0x00,0x00,0x00,  /* 00000000    "GTDTP..." */
438     0x01,0xF1,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
439     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
440     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
441     0x23,0x06,0x11,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "#..  ...." */
442     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000028    "........" */
443     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
444     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
445     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
446     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000048    "........" */
447 };

449 const unsigned char TemplateHest[] =
450 {
451     0x48,0x45,0x53,0x54,0xD4,0x01,0x00,0x00,  /* 00000000    "HEST...." */
452     0x01,0x20,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ". INTEL " */
453     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
454     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
455     0x28,0x05,0x10,0x20,0x04,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
```



```
456     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,  /* 00000028    "........" */
457     0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000030    "........" */
458     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
459     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
460     0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
461     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
462     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
463     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
464     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000068    "........" */
465     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000070    "........" */
466     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
467     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
468     0x01,0x00,0x01,0x00,0x00,0x00,0x00,0x01,  /* 00000088    "........" */
469     0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000090    "........" */
470     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
471     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
472     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A8    "........" */
473     0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x00,  /* 000000B0    "........" */
474     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
475     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C0    "........" */
476     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
477     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 000000D0    "........" */
478     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000D8    "........" */
479     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
480     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
481     0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x01,  /* 000000F0    "........" */
482     0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 000000F8    "........" */
483     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
484     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000108    "........" */
485     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000110    "........" */
486     0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,  /* 00000118    "........" */
487     0x00,0x00,0x00,0x00,0x01,0x01,0x00,0x00,  /* 00000120    "........" */
488     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000128    "........" */
489     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000130    "........" */
490     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000138    "........" */
491     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000140    "........" */
492     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000148    "........" */
493     0x00,0x00,0x00,0x00,0x09,0x00,0x02,0x00,  /* 00000150    "........" */
494     0xFF,0xFF,0x00,0x01,0x01,0x00,0x00,0x00,  /* 00000158    "........" */
495     0x01,0x00,0x00,0x00,0x00,0x10,0x00,0x00,  /* 00000160    "........" */
496     0x00,0x40,0x00,0x04,0x00,0x00,0x00,0x00,  /* 00000168    ".@......" */
497     0x00,0x00,0x00,0x00,0x03,0x1C,0x00,0x00,  /* 00000170    "........" */
498     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000178    "........" */
499     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000180    "........" */
500     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000188    "........" */
501     0x00,0x10,0x00,0x00,0x09,0x00,0x03,0x00,  /* 00000190    "........" */
502     0x00,0x00,0x00,0x01,0x01,0x00,0x00,0x00,  /* 00000198    "........" */
503     0x01,0x00,0x00,0x00,0x00,0x10,0x00,0x00,  /* 000001A0    "........" */
504     0x00,0x40,0x00,0x04,0x00,0x00,0x00,0x00,  /* 000001A8    ".@......" */
505     0x00,0x00,0x00,0x00,0x04,0x1C,0x00,0x00,  /* 000001B0    "........" */
506     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001B8    "........" */
507     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001C0    "........" */
508     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000001C8    "........" */
509     0x00,0x10,0x00,0x00                       /* 000001D0    "...."     */
510 };

512 const unsigned char TemplateHpet[] =
513 {
514     0x48,0x50,0x45,0x54,0x38,0x00,0x00,0x00,  /* 00000000    "HPET8..." */
515     0x01,0x09,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
516     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
517     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
518     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
519     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
520     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000030    "........" */
521 };
```

```
const unsigned char TemplateIvrs[] =
{
    0x49,0x56,0x52,0x53,0xBC,0x00,0x00,0x00,  /* 00000000    "IVRS...." */
    0x01,0x87,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x10,0x14,0x34,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "..4....." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x00,  /* 00000048    "....@..." */
    0x00,0x00,0x00,0x00,0x42,0x00,0x00,0x00,  /* 00000050    "....B..." */
    0x00,0x00,0x00,0x00,0x48,0x00,0x00,0x00,  /* 00000058    "....H..." */
    0x00,0x00,0x00,0x00,0x20,0x08,0x20,0x00,  /* 00000060    ".... . ." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000070    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x21,0x04,0x20,0x00,  /* 00000080    "....!. ." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000088    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000090    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x00,0x00,0x10,0x14,0x18,0x00,  /* 000000A0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B0    "........" */
    0x00,0x00,0x00,0x00                       /* 000000B8    "...."     */
};

/* MADT with ACPI 5.0 subtables */

const unsigned char TemplateMadt[] =
{
    0x41,0x50,0x49,0x43,0xF6,0x00,0x00,0x00,  /* 00000000    "APIC...." */
    0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x23,0x06,0x11,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "#.. ...." */
    0x01,0x00,0x00,0x00,0x00,0x08,0x00,0x00,  /* 00000028    "........" */
    0x01,0x00,0x00,0x00,0x01,0x0C,0x01,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x02,0x0A,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x03,0x08,0x0D,0x00,0x01,0x00,  /* 00000048    "........" */
    0x00,0x00,0x04,0x06,0x00,0x05,0x00,0x01,  /* 00000050    "........" */
    0x05,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x06,0x10,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x07,0x16,0x00,0x00,  /* 00000070    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x5C,0x43,0x50,0x55,  /* 00000080    "....\CPU" */
    0x30,0x00,0x08,0x10,0x05,0x00,0x00,0x00,  /* 00000088    "0......." */
    0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,  /* 00000090    "........" */
    0x00,0x00,0x09,0x10,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0x00,0x00,0x0A,0x0C,0x05,0x00,0x00,0x00,  /* 000000A8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x0B,0x28,  /* 000000B0    ".......(" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
    0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,  /* 000000C0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000D0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x0C,0x18,  /* 000000D8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00             /* 000000F0    "......"   */
};
```



```
const unsigned char TemplateMcfg[] =
{
    0x4D,0x43,0x46,0x47,0x3C,0x00,0x00,0x00,  /* 00000000    "MCFG<..." */
    0x01,0x19,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00                  /* 00000038    "...."     */
};

const unsigned char TemplateMchi[] =
{
    0x4D,0x43,0x48,0x49,0x45,0x00,0x00,0x00,  /* 00000000    "MCHIE..." */
    0x01,0xE4,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x15,0x07,0x00,0x02,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x02,0x08,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00                  /* 00000040    "....."    */
};

const unsigned char TemplateMpst[] =
{
    0x4D,0x50,0x53,0x54,0xB6,0x00,0x00,0x00,  /* 00000000    "MPST...." */
    0x01,0x77,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".wINTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x13,0x09,0x12,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x00,  /* 00000040    "........" */
    0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
    0x00,0x00,0x01,0x00,0x02,0x00,0x00,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,  /* 00000068    "........" */
    0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,  /* 00000070    "........" */
    0x00,0x00,0x02,0x00,0x00,0x00,0x01,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000088    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000090    "........" */
    0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A8    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00             /* 000000B0    "......"   */
};

const unsigned char TemplateMsct[] =
{
    0x4D,0x53,0x43,0x54,0x90,0x00,0x00,0x00,  /* 00000000    "MSCT...." */
    0x01,0xB7,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x38,0x00,0x00,0x00,  /* 00000020    "(.. 8..." */
    0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0x0F,0x00,0x00,  /* 00000030    "........" */
    0x01,0x16,0x00,0x00,0x00,0x00,0x03,0x00,  /* 00000038    "........" */
    0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x40,0x00,0x00,0x00,0x01,0x16,  /* 00000048    "..@....." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
```

```
    0x00,0x00,0x00,0x00,0x01,0x16,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000070    "........" */
    0x00,0x00,0x01,0x16,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000088    "........" */
};

const unsigned char TemplateMtmr[] =
{
    0x4D,0x54,0x4D,0x52,0x4C,0x00,0x00,0x00,  /* 00000000    "MTMRL..." */
    0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x03,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x17,0x01,0x13,0x20,0x00,0x20,0x00,0x03,  /* 00000020    "... . .." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x20,0x00,0x03,0x00,0x00,0x00,0x00,  /* 00000038    ". ......" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x00,0x00                       /* 00000048    "...."     */
};

const unsigned char TemplatePcct[] =
{
    0x50,0x43,0x43,0x54,0xAC,0x00,0x00,0x00,  /* 00000000    "PCCT...." */
    0x01,0x97,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x3E,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    ".>......" */
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,  /* 00000038    "........" */
    0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,  /* 00000040    """"""""" */
    0x01,0x32,0x00,0x03,0x33,0x33,0x33,0x33,  /* 00000048    ".2..3333" */
    0x33,0x33,0x33,0x33,0x44,0x44,0x44,0x44,  /* 00000050    "3333DDDD" */
    0x44,0x44,0x44,0x44,0x55,0x55,0x55,0x55,  /* 00000058    "DDDDUUUU" */
    0x55,0x55,0x55,0x55,0x66,0x66,0x66,0x66,  /* 00000060    "UUUUffff" */
    0x77,0x77,0x77,0x77,0x88,0x88,0x00,0x3E,  /* 00000068    "wwww...>" */
    0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFF,  /* 00000070    "........" */
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xEE,0xEE,  /* 00000078    "........" */
    0xEE,0xEE,0xEE,0xEE,0xEE,0xEE,0x01,0x32,  /* 00000080    ".......2" */
    0x00,0x03,0xDD,0xDD,0xDD,0xDD,0xDD,0xDD,  /* 00000088    "........" */
    0xDD,0xDD,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,  /* 00000090    "........" */
    0xCC,0xCC,0xBB,0xBB,0xBB,0xBB,0xBB,0xBB,  /* 00000098    "........" */
    0xBB,0xBB,0xAA,0xAA,0xAA,0xAA,0x99,0x99,  /* 000000A0    "........" */
    0x99,0x99,0x88,0x88                       /* 000000A8    "...."     */
};

const unsigned char TemplatePmtt[] =
{
    0x50,0x4D,0x54,0x54,0xB4,0x00,0x00,0x00,  /* 00000000    "PMTT...." */
    0x01,0x3A,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".:INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x26,0x08,0x11,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "&.. ...." */
    0x00,0x00,0x80,0x00,0x01,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x54,0x00,  /* 00000030    "......T." */
    0x05,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
    0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
    0x02,0x00,0x14,0x00,0x02,0x00,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x02,0x00,0x14,0x00,  /* 00000070    "........" */
    0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
```



```
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
    0x01,0x00,0x20,0x00,0x01,0x00,0x00,0x00,  /* 00000088    ".. ....." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000090    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0x00,0x00,0x0C,0x00,0x01,0x00,0x00,0x00,  /* 000000A8    "........" */
    0x00,0x00,0x00,0x00                       /* 000000B0    "....."    */
};

const unsigned char TemplateRsdp[] =
{
    0x52,0x53,0x44,0x20,0x50,0x54,0x52,0x20,  /* 00000000    "RSD PTR " */
    0x43,0x49,0x4E,0x54,0x45,0x4C,0x20,0x02,  /* 00000008    "CINTEL ." */
    0x00,0x00,0x00,0x00,0x24,0x00,0x00,0x00,  /* 00000010    "....$..." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000018    "........" */
    0xDC,0x00,0x00,0x00                       /* 00000020    "...."     */
};

const unsigned char TemplateRsdt[] =
{
    0x52,0x53,0x44,0x54,0x44,0x00,0x00,0x00,  /* 00000000    "RSDTD..." */
    0x01,0xB1,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x10,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x20,0x00,0x00,0x00,0x30,0x00,0x00,0x00,  /* 00000028    " ...0..." */
    0x40,0x00,0x00,0x00,0x50,0x00,0x00,0x00,  /* 00000030    "@...P..." */
    0x60,0x00,0x00,0x00,0x70,0x00,0x00,0x00,  /* 00000038    "`...p..." */
    0x80,0x00,0x00,0x00                       /* 00000040    "...."     */
};

const unsigned char TemplateS3pt[] =
{
    0x53,0x33,0x50,0x54,0x34,0x00,0x00,0x00,  /* 00000000    "S3PT4..." */
    0x00,0x00,0x18,0x01,0x00,0x00,0x00,0x00,  /* 00000008    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000010    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000018    "........" */
    0x01,0x00,0x14,0x01,0x00,0x00,0x00,0x00,  /* 00000020    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00                       /* 00000030    "...."     */
};

const unsigned char TemplateSbst[] =
{
    0x53,0x42,0x53,0x54,0x30,0x00,0x00,0x00,  /* 00000000    "SBST0..." */
    0x01,0x06,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000028    "........" */
};

const unsigned char TemplateSlic[] =
{
    0x53,0x4C,0x49,0x43,0x76,0x01,0x00,0x00,  /* 00000000    "SLICv..." */
    0x01,0x07,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x11,0x02,0x11,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x9C,0x00,0x00,0x00,0x06,0x02,0x00,0x00,  /* 00000028    "........" */
    0x00,0x24,0x00,0x00,0x52,0x53,0x41,0x31,  /* 00000030    ".$..RSA1" */
    0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
```

```
786     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
787     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
788     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000070    "........" */
789     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
790     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
791     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000088    "........" */
792     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000090    "........" */
793     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
794     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
795     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A8    "........" */
796     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B0    "........" */
797     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
798     0x01,0x00,0x00,0x00,0xB6,0x00,0x00,0x00,  /* 000000C0    "........" */
799     0x00,0x00,0x02,0x00,0x49,0x4E,0x54,0x45,  /* 000000C8    "....INTE" */
800     0x4C,0x20,0x54,0x45,0x4D,0x50,0x4C,0x41,  /* 000000D0    "L TEMPLA" */
801     0x54,0x45,0x57,0x49,0x4E,0x44,0x4F,0x57,  /* 000000D8    "TEWINDOW" */
802     0x53,0x20,0x01,0x00,0x02,0x00,0x00,0x00,  /* 000000E0    "S ......" */
803     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
804     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F0    "........" */
805     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
806     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
807     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000108    "........" */
808     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000110    "........" */
809     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000118    "........" */
810     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000120    "........" */
811     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000128    "........" */
812     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000130    "........" */
813     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000138    "........" */
814     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000140    "........" */
815     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000148    "........" */
816     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000150    "........" */
817     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000158    "........" */
818     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000160    "........" */
819     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000168    "........" */
820     0x00,0x00,0x00,0x00,0x00,0x00             /* 00000170    "......"   */
821 };

823 const unsigned char TemplateSlit[] =
824 {
825     0x53,0x4C,0x49,0x54,0xBC,0x01,0x00,0x00,  /* 00000000    "SLIT...." */
826     0x01,0x00,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
827     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
828     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
829     0x16,0x03,0x11,0x20,0x14,0x00,0x00,0x00,  /* 00000020    "... ...." */
830     0x00,0x00,0x00,0x00,0x0A,0x10,0x16,0x17,  /* 00000028    "........" */
831     0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,  /* 00000030    "........" */
832     0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,  /* 00000038    " !"#$%&'" */
833     0x10,0x0A,0x15,0x16,0x17,0x18,0x19,0x1A,  /* 00000040    "........" */
834     0x1B,0x1C,0x1D,0x1E,0x1F,0x20,0x21,0x22,  /* 00000048    "..... !"" */
835     0x23,0x24,0x25,0x26,0x16,0x15,0x0A,0x10,  /* 00000050    "#$%&...." */
836     0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,  /* 00000058    "........" */
837     0x1E,0x1F,0x20,0x21,0x22,0x23,0x24,0x25,  /* 00000060    ".. !"#$%" */
838     0x17,0x16,0x10,0x0A,0x15,0x16,0x17,0x18,  /* 00000068    "........" */
839     0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,0x20,  /* 00000070    "........ " */
840     0x21,0x22,0x23,0x24,0x18,0x17,0x16,0x15,  /* 00000078    "!"#$...." */
841     0x0A,0x10,0x16,0x17,0x18,0x19,0x1A,0x1B,  /* 00000080    "........" */
842     0x1C,0x1D,0x1E,0x1F,0x20,0x21,0x22,0x23,  /* 00000088    ".... !"#" */
843     0x19,0x18,0x17,0x16,0x10,0x0A,0x15,0x16,  /* 00000090    "........" */
844     0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,  /* 00000098    "........" */
845     0x1F,0x20,0x21,0x22,0x1A,0x19,0x18,0x17,  /* 000000A0    ". !"...." */
846     0x16,0x15,0x0A,0x10,0x16,0x17,0x18,0x19,  /* 000000A8    "........" */
847     0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,0x20,0x21,  /* 000000B0    "...... !" */
848     0x1B,0x1A,0x19,0x18,0x17,0x16,0x10,0x0A,  /* 000000B8    "........" */
849     0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,  /* 000000C0    "........" */
850     0x1D,0x1E,0x1F,0x20,0x1C,0x1B,0x1A,0x19,  /* 000000C8    "... ...." */
851     0x18,0x17,0x16,0x15,0x0A,0x10,0x16,0x17,  /* 000000D0    "........" */
```



```
852     0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,  /* 000000D8    "........" */
853     0x1D,0x1C,0x1B,0x1A,0x19,0x18,0x17,0x16,  /* 000000E0    "........" */
854     0x10,0x0A,0x15,0x16,0x17,0x18,0x19,0x1A,  /* 000000E8    "........" */
855     0x1B,0x1C,0x1D,0x1E,0x1E,0x1D,0x1C,0x1B,  /* 000000F0    "........" */
856     0x1A,0x19,0x18,0x17,0x16,0x15,0x0A,0x10,  /* 000000F8    "........" */
857     0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,  /* 00000100    "........" */
858     0x1F,0x1E,0x1D,0x1C,0x1B,0x1A,0x19,0x18,  /* 00000108    "........" */
859     0x17,0x16,0x10,0x0A,0x15,0x16,0x17,0x18,  /* 00000110    "........" */
860     0x19,0x1A,0x1B,0x1C,0x20,0x1F,0x1E,0x1D,  /* 00000118    ".... ..." */
861     0x1C,0x1B,0x1A,0x19,0x18,0x17,0x16,0x15,  /* 00000120    "........" */
862     0x0A,0x10,0x16,0x17,0x18,0x19,0x1A,0x1B,  /* 00000128    "........" */
863     0x21,0x20,0x1F,0x1E,0x1D,0x1C,0x1B,0x1A,  /* 00000130    "! ......" */
864     0x19,0x18,0x17,0x16,0x10,0x0A,0x15,0x16,  /* 00000138    "........" */
865     0x17,0x18,0x19,0x1A,0x22,0x21,0x20,0x1F,  /* 00000140    "....\"! ." */
866     0x1E,0x1D,0x1C,0x1B,0x1A,0x19,0x18,0x17,  /* 00000148    "........" */
867     0x16,0x15,0x0A,0x10,0x16,0x17,0x18,0x19,  /* 00000150    "........" */
868     0x23,0x22,0x21,0x20,0x1F,0x1E,0x1D,0x1C,  /* 00000158    "#\"! ...." */
869     0x1B,0x1A,0x19,0x18,0x17,0x16,0x10,0x0A,  /* 00000160    "........" */
870     0x15,0x16,0x17,0x18,0x24,0x23,0x22,0x21,  /* 00000168    "....$#\"!" */
871     0x20,0x1F,0x1E,0x1D,0x1C,0x1B,0x1A,0x19,  /* 00000170    " ......." */
872     0x18,0x17,0x16,0x15,0x0A,0x10,0x16,0x17,  /* 00000178    "........" */
873     0x25,0x24,0x23,0x22,0x21,0x20,0x1F,0x1E,  /* 00000180    "%$#\"! .." */
874     0x1D,0x1C,0x1B,0x1A,0x19,0x18,0x17,0x16,  /* 00000188    "........" */
875     0x10,0x0A,0x15,0x16,0x26,0x25,0x24,0x23,  /* 00000190    "....&%$#" */
876     0x22,0x21,0x20,0x1F,0x1E,0x1D,0x1C,0x1B,  /* 00000198    "\"! ....." */
877     0x1A,0x19,0x18,0x17,0x16,0x15,0x0A,0x10,  /* 000001A0    "........" */
878     0x27,0x26,0x25,0x24,0x23,0x22,0x21,0x20,  /* 000001A8    "'&%$#\"! " */
879     0x1F,0x1E,0x1D,0x1C,0x1B,0x1A,0x19,0x18,  /* 000001B0    "........" */
880     0x17,0x16,0x10,0x0A                       /* 000001B8    "....""    */
881 };

883 const unsigned char TemplateSpcr[] =
884 {
885     0x53,0x50,0x43,0x52,0x50,0x00,0x00,0x00,  /* 00000000    "SPCRP..." */
886     0x01,0xE3,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
887     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
888     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
889     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
890     0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
891     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
892     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
893     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
894     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000048    "........" */
895 };

897 const unsigned char TemplateSpmi[] =
898 {
899     0x53,0x50,0x4D,0x49,0x41,0x00,0x00,0x00,  /* 00000000    "SPMIA..." */
900     0x04,0xED,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
901     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
902     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
903     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
904     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
905     0x00,0x08,0x00,0x01,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
906     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
907     0x00                                      /* 00000040    "."        */
908 };

910 const unsigned char TemplateSrat[] =
911 {
912     0x53,0x52,0x41,0x54,0x80,0x00,0x00,0x00,  /* 00000000    "SRAT...." */
913     0x03,0x5A,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".ZINTEL " */
914     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
915     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
916     0x28,0x05,0x10,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
917     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
```

```
    0x00,0x10,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x01,0x28,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    ".(......" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
    0x00,0xFC,0x09,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0x02,0x18,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000070    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000078    "........" */
};

const unsigned char TemplateTcpa[] =
{
    0x54,0x43,0x50,0x41,0x32,0x00,0x00,0x00,  /* 00000000    "TCPA2..." */
    0x01,0x67,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".gINTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x80,0x31,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    ".1..INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00                                 /* 00000030    ".."       */
};

const unsigned char TemplateTpm2[] =
{
    0x54,0x50,0x4D,0x32,0x34,0x00,0x00,0x00,  /* 00000000    "TPM24..." */
    0x03,0x42,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".BINTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x14,0x11,0x12,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x77,0x66,0x55,0x44,0x33,0x22,0x11,0x00,  /* 00000028    "wfUD3".." */
    0x01,0x00,0x00,0x00                       /* 00000030    "....      */
};

const unsigned char TemplateUefi[] =
{
    0x55,0x45,0x46,0x49,0x36,0x00,0x00,0x00,  /* 00000000    "UEFI6..." */
    0x01,0x9B,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x01,0x02,0x03,  /* 00000020    "(.. ...." */
    0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,  /* 00000028    "........" */
    0x0C,0x0D,0x0E,0x0F,0x00,0x00             /* 00000030    "......"   */
};

const unsigned char TemplateVrtc[] =
{
    0x56,0x52,0x54,0x43,0x44,0x00,0x00,0x00,  /* 00000000    "VRTCD..." */
    0x01,0xEF,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x03,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x17,0x01,0x13,0x20,0x00,0x08,0x00,0x00,  /* 00000020    "... ...." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00                       /* 00000040    "....      */
};

const unsigned char TemplateWaet[] =
{
    0x57,0x41,0x45,0x54,0x28,0x00,0x00,0x00,  /* 00000000    "WAET(..." */
    0x01,0x19,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00   /* 00000020    "(.. ...." */
};
```



```
const unsigned char TemplateWdat[] =
{
    0x57,0x44,0x41,0x54,0x5C,0x00,0x00,0x00,  /* 00000000    "WDAT\..." */
    0x01,0xE3,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x20,0x00,0x00,0x00,  /* 00000020    "(..  ..." */
    0xFF,0x00,0xFF,0xFF,0xFF,0x00,0x00,0x00,  /* 00000028    "........" */
    0x58,0x02,0x00,0x00,0xFF,0x03,0x00,0x00,  /* 00000030    "X......." */
    0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x0E,0x00,0x00,0x00,0x01,0x02,0x00,0x00,  /* 00000040    "........" */
    0x01,0x10,0x00,0x02,0x60,0x04,0x00,0x00,  /* 00000048    "....'..." */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000050    "........" */
    0x01,0x00,0x00,0x00                       /* 00000058    "....      */
};

const unsigned char TemplateWddt[] =
{
    0x57,0x44,0x44,0x54,0x40,0x00,0x00,0x00,  /* 00000000    "WDDT@..." */
    0x01,0x00,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x01,0xFF,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000038    "........" */
};

const unsigned char TemplateWdrt[] =
{
    0x57,0x44,0x52,0x54,0x47,0x00,0x00,0x00,  /* 00000000    "WDRTG..." */
    0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x00,0x20,0x00,0x00,  /* 00000020    "(.. . .." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    ". ......" */
    0x00,0x00,0x00,0x00,0xFF,0xFF,0xFF,0xFF,  /* 00000038    "........" */
    0x00,0x00,0x00,0x00,0xFF,0xFF,0x00        /* 00000040    "......."  */
};

const unsigned char TemplateXsdt[] =
{
    0x58,0x53,0x44,0x54,0x64,0x00,0x00,0x00,  /* 00000000    "XSDTd..." */
    0x01,0x8B,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x28,0x05,0x10,0x20,0x10,0x00,0x00,0x00,  /* 00000020    "(.. ...." */
    0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,  /* 00000028    ".... ..." */
    0x00,0x00,0x00,0x00,0x30,0x00,0x00,0x00,  /* 00000030    "....0..." */
    0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x00,  /* 00000038    "....@..." */
    0x00,0x00,0x00,0x00,0x50,0x00,0x00,0x00,  /* 00000040    "....P..." */
    0x00,0x00,0x00,0x00,0x60,0x00,0x00,0x00,  /* 00000048    "....'..." */
    0x00,0x00,0x00,0x00,0x70,0x00,0x00,0x00,  /* 00000050    "....p..." */
    0x00,0x00,0x00,0x00,0x80,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00                       /* 00000060    "....      */
};

#endif
```

```
**********************************************************
   21549 Thu Dec 26 13:48:39 2013
new/usr/src/common/acpica/compiler/dtutils.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dtutils.c - Utility routines for the data table compiler
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DTUTILS_C__

#include "aslcompiler.h"
#include "dtcompiler.h"
#include "actables.h"

#define _COMPONENT          DT_COMPILER
        ACPI_MODULE_NAME    ("dtutils")

/* Local prototypes */

static void
DtSum (
    DT_SUBTABLE             *Subtable,
    void                    *Context,
    void                    *ReturnValue);
```



```
/******************************************************************************
 *
 * FUNCTION:    DtError
 *
 * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
 *              MessageId           - Index into global message buffer
 *              Op                  - Parse node where error happened
 *              ExtraMessage        - additional error message
 *
 * RETURN:      None
 *
 * DESCRIPTION: Common error interface for data table compiler
 *
 *****************************************************************************/

void
DtError (
    UINT8                   Level,
    UINT8                   MessageId,
    DT_FIELD                *FieldObject,
    char                    *ExtraMessage)
{

    /* Check if user wants to ignore this exception */

    if (AslIsExceptionDisabled (Level, MessageId))
    {
        return;
    }

    if (FieldObject)
    {
        AslCommonError (Level, MessageId,
            FieldObject->Line,
            FieldObject->Line,
            FieldObject->ByteOffset,
            FieldObject->Column,
            Gbl_Files[ASL_FILE_INPUT].Filename, ExtraMessage);
    }
    else
    {
        AslCommonError (Level, MessageId, 0,
            0, 0, 0, 0, ExtraMessage);
    }
}


/******************************************************************************
 *
 * FUNCTION:    DtNameError
 *
 * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
 *              MessageId           - Index into global message buffer
 *              Op                  - Parse node where error happened
 *              ExtraMessage        - additional error message
 *
 * RETURN:      None
 *
 * DESCRIPTION: Error interface for named objects
 *
 *****************************************************************************/

void
DtNameError (
```

```
126     UINT8                   Level,
127     UINT8                   MessageId,
128     DT_FIELD                *FieldObject,
129     char                    *ExtraMessage)
130 {

132     switch (Level)
133     {
134     case ASL_WARNING2:
135     case ASL_WARNING3:

137         if (Gbl_WarningLevel < Level)
138         {
139             return;
140         }
141         break;

143     default:

145         break;
146     }

148     if (FieldObject)
149     {
150         AslCommonError (Level, MessageId,
151             FieldObject->Line,
152             FieldObject->Line,
153             FieldObject->ByteOffset,
154             FieldObject->NameColumn,
155             Gbl_Files[ASL_FILE_INPUT].Filename, ExtraMessage);
156     }
157     else
158     {
159         AslCommonError (Level, MessageId, 0,
160             0, 0, 0, 0, ExtraMessage);
161     }
162 }


165 /*******************************************************************************
166  *
167  * FUNCTION:    DtFatal
168  *
169  * PARAMETERS:  None
170  *
171  * RETURN:      None
172  *
173  * DESCRIPTION: Dump the error log and abort the compiler. Used for serious
174  *              compile or I/O errors
175  *
176  ******************************************************************************/

178 void
179 DtFatal (
180     UINT8                   MessageId,
181     DT_FIELD                *FieldObject,
182     char                    *ExtraMessage)
183 {

185     DtError (ASL_ERROR, MessageId, FieldObject, ExtraMessage);

187 /*
188  * TBD: remove this entire function, DtFatal
189  *
190  * We cannot abort the compiler on error, because we may be compiling a
191  * list of files. We must move on to the next file.
```



```
192  */
193 #ifdef __OBSOLETE
194     CmCleanupAndExit ();
195     exit (1);
196 #endif
197 }


200 /*******************************************************************************
201  *
202  * FUNCTION:    DtStrtoul64
203  *
204  * PARAMETERS:  String              - Null terminated string
205  *              ReturnInteger       - Where the converted integer is returned
206  *
207  * RETURN:      Status
208  *
209  * DESCRIPTION: Simple conversion of a string hex integer constant to unsigned
210  *              value. Assumes no leading "0x" for the constant.
211  *
212  * Portability note: The reason this function exists is because a 64-bit
213  * sscanf is not available in all environments.
214  *
215  ******************************************************************************/

217 ACPI_STATUS
218 DtStrtoul64 (
219     char                    *String,
220     UINT64                  *ReturnInteger)
221 {
222     char                    *ThisChar = String;
223     UINT32                  ThisDigit;
224     UINT64                  ReturnValue = 0;
225     int                     DigitCount = 0;


228     /* Skip over any white space in the buffer */

230     while ((*ThisChar == ' ') || (*ThisChar == '\t'))
231     {
232         ThisChar++;
233     }

235     /* Skip leading zeros */

237     while ((*ThisChar) == '0')
238     {
239         ThisChar++;
240     }

242     /* Convert character-by-character */

244     while (*ThisChar)
245     {
246         if (ACPI_IS_DIGIT (*ThisChar))
247         {
248             /* Convert ASCII 0-9 to Decimal value */

250             ThisDigit = ((UINT8) *ThisChar) - '0';
251         }
252         else /* Letter */
253         {
254             ThisDigit = (UINT32) ACPI_TOUPPER (*ThisChar);
255             if (!ACPI_IS_XDIGIT ((char) ThisDigit))
256             {
257                 /* Not A-F */
```

```
259                 return (AE_BAD_CHARACTER);
260             }

262             /* Convert ASCII Hex char (A-F) to value */

264             ThisDigit = (ThisDigit - 'A') + 10;
265         }

267         /* Insert the 4-bit hex digit */

269         ReturnValue <<= 4;
270         ReturnValue += ThisDigit;

272         ThisChar++;
273         DigitCount++;
274         if (DigitCount > 16)
275         {
276             /* Value is too large (> 64 bits/8 bytes/16 hex digits) */

278             return (AE_LIMIT);
279         }
280     }

282     *ReturnInteger = ReturnValue;
283     return (AE_OK);
284 }


287 /*******************************************************************************
288  *
289  * FUNCTION:    DtGetFileSize
290  *
291  * PARAMETERS:  Handle              - Open file handler
292  *
293  * RETURN:      Current file size
294  *
295  * DESCRIPTION: Get the current size of a file. Seek to the EOF and get the
296  *              offset. Seek back to the original location.
297  *
298  ******************************************************************************/

300 UINT32
301 DtGetFileSize (
302     FILE                    *Handle)
303 {
304     int                     CurrentOffset;
305     int                     LastOffset;


308     CurrentOffset = ftell (Handle);
309     fseek (Handle, 0, SEEK_END);
310     LastOffset = ftell (Handle);
311     fseek (Handle, CurrentOffset, SEEK_SET);

313     return ((UINT32) LastOffset);
314 }


317 /*******************************************************************************
318  *
319  * FUNCTION:    DtGetFieldValue
320  *
321  * PARAMETERS:  Field               - Current field list pointer
322  *
323  * RETURN:      Field value
```



```
324  *
325  * DESCRIPTION: Get field value
326  *
327  ******************************************************************************/

329 char *
330 DtGetFieldValue (
331     DT_FIELD                *Field)
332 {
333     if (!Field)
334     {
335         return (NULL);
336     }

338     return (Field->Value);
339 }


342 /*******************************************************************************
343  *
344  * FUNCTION:    DtGetFieldType
345  *
346  * PARAMETERS:  Info                - Data table info
347  *
348  * RETURN:      Field type
349  *
350  * DESCRIPTION: Get field type
351  *
352  ******************************************************************************/

354 UINT8
355 DtGetFieldType (
356     ACPI_DMTABLE_INFO       *Info)
357 {
358     UINT8                   Type;


361     /* DT_FLAG means that this is the start of a block of flag bits */
362     /* TBD - we can make these a separate opcode later */

364     if (Info->Flags & DT_FLAG)
365     {
366         return (DT_FIELD_TYPE_FLAGS_INTEGER);
367     }

369     /* Type is based upon the opcode for this field in the info table */

371     switch (Info->Opcode)
372     {
373     case ACPI_DMT_FLAG0:
374     case ACPI_DMT_FLAG1:
375     case ACPI_DMT_FLAG2:
376     case ACPI_DMT_FLAG3:
377     case ACPI_DMT_FLAG4:
378     case ACPI_DMT_FLAG5:
379     case ACPI_DMT_FLAG6:
380     case ACPI_DMT_FLAG7:
381     case ACPI_DMT_FLAGS0:
382     case ACPI_DMT_FLAGS1:
383     case ACPI_DMT_FLAGS2:
384     case ACPI_DMT_FLAGS4:

386         Type = DT_FIELD_TYPE_FLAG;
387         break;

389     case ACPI_DMT_NAME4:
```

```
390     case ACPI_DMT_SIG:
391     case ACPI_DMT_NAME6:
392     case ACPI_DMT_NAME8:
393     case ACPI_DMT_STRING:

395         Type = DT_FIELD_TYPE_STRING;
396         break;

398     case ACPI_DMT_BUFFER:
399     case ACPI_DMT_BUF7:
400     case ACPI_DMT_BUF10:
401     case ACPI_DMT_BUF16:
402     case ACPI_DMT_BUF128:
403     case ACPI_DMT_PCI_PATH:

405         Type = DT_FIELD_TYPE_BUFFER;
406         break;

408     case ACPI_DMT_GAS:
409     case ACPI_DMT_HESTNTFY:

411         Type = DT_FIELD_TYPE_INLINE_SUBTABLE;
412         break;

414     case ACPI_DMT_UNICODE:

416         Type = DT_FIELD_TYPE_UNICODE;
417         break;

419     case ACPI_DMT_UUID:

421         Type = DT_FIELD_TYPE_UUID;
422         break;

424     case ACPI_DMT_DEVICE_PATH:

426         Type = DT_FIELD_TYPE_DEVICE_PATH;
427         break;

429     case ACPI_DMT_LABEL:

431         Type = DT_FIELD_TYPE_LABEL;
432         break;

434     default:

436         Type = DT_FIELD_TYPE_INTEGER;
437         break;
438     }

440     return (Type);
441 }


444 /*******************************************************************************
445  *
446  * FUNCTION:    DtGetBufferLength
447  *
448  * PARAMETERS:  Buffer              - List of integers,
449  *                                    for example "10 3A 4F 2E"
450  *
451  * RETURN:      Count of integer
452  *
453  * DESCRIPTION: Get length of bytes needed to store the integers
454  *
455  ******************************************************************************/
```



```
457 UINT32
458 DtGetBufferLength (
459     char                    *Buffer)
460 {
461     UINT32                  ByteLength = 0;


464     while (*Buffer)
465     {
466         if (*Buffer == ' ')
467         {
468             ByteLength++;

470             while (*Buffer == ' ')
471             {
472                 Buffer++;
473             }
474         }

476         Buffer++;
477     }

479     return (++ByteLength);
480 }


483 /*******************************************************************************
484  *
485  * FUNCTION:    DtGetFieldLength
486  *
487  * PARAMETERS:  Field               - Current field
488  *              Info                - Data table info
489  *
490  * RETURN:      Field length
491  *
492  * DESCRIPTION: Get length of bytes needed to compile the field
493  *
494  * Note: This function must remain in sync with AcpiDmDumpTable.
495  *
496  ******************************************************************************/

498 UINT32
499 DtGetFieldLength (
500     DT_FIELD                *Field,
501     ACPI_DMTABLE_INFO       *Info)
502 {
503     UINT32                  ByteLength = 0;
504     char                    *Value;


507     /* Length is based upon the opcode for this field in the info table */

509     switch (Info->Opcode)
510     {
511     case ACPI_DMT_FLAG0:
512     case ACPI_DMT_FLAG1:
513     case ACPI_DMT_FLAG2:
514     case ACPI_DMT_FLAG3:
515     case ACPI_DMT_FLAG4:
516     case ACPI_DMT_FLAG5:
517     case ACPI_DMT_FLAG6:
518     case ACPI_DMT_FLAG7:
519     case ACPI_DMT_FLAGS0:
520     case ACPI_DMT_FLAGS1:
521     case ACPI_DMT_FLAGS2:
```

```
522     case ACPI_DMT_FLAGS4:
523     case ACPI_DMT_LABEL:
524     case ACPI_DMT_EXTRA_TEXT:

526         ByteLength = 0;
527         break;

529     case ACPI_DMT_UINT8:
530     case ACPI_DMT_CHKSUM:
531     case ACPI_DMT_SPACEID:
532     case ACPI_DMT_ACCWIDTH:
533     case ACPI_DMT_IVRS:
534     case ACPI_DMT_MADT:
535     case ACPI_DMT_PCCT:
536     case ACPI_DMT_PMTT:
537     case ACPI_DMT_SRAT:
538     case ACPI_DMT_ASF:
539     case ACPI_DMT_HESTNTYP:
540     case ACPI_DMT_FADTPM:
541     case ACPI_DMT_EINJACT:
542     case ACPI_DMT_EINJINST:
543     case ACPI_DMT_ERSTACT:
544     case ACPI_DMT_ERSTINST:

546         ByteLength = 1;
547         break;

549     case ACPI_DMT_UINT16:
550     case ACPI_DMT_DMAR:
551     case ACPI_DMT_HEST:
552     case ACPI_DMT_PCI_PATH:

554         ByteLength = 2;
555         break;

557     case ACPI_DMT_UINT24:

559         ByteLength = 3;
560         break;

562     case ACPI_DMT_UINT32:
563     case ACPI_DMT_NAME4:
564     case ACPI_DMT_SLIC:
565     case ACPI_DMT_SIG:

567         ByteLength = 4;
568         break;

570     case ACPI_DMT_UINT40:

572         ByteLength = 5;
573         break;

575     case ACPI_DMT_UINT48:
576     case ACPI_DMT_NAME6:

578         ByteLength = 6;
579         break;

581     case ACPI_DMT_UINT56:
582     case ACPI_DMT_BUF7:

584         ByteLength = 7;
585         break;

587     case ACPI_DMT_UINT64:
```



```
588     case ACPI_DMT_NAME8:

590         ByteLength = 8;
591         break;

593     case ACPI_DMT_STRING:

595         Value = DtGetFieldValue (Field);
596         if (Value)
597         {
598             ByteLength = ACPI_STRLEN (Value) + 1;
599         }
600         else
601         {   /* At this point, this is a fatal error */

603             sprintf (MsgBuffer, "Expected \"%s\"", Info->Name);
604             DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, MsgBuffer);
605             return (0);
606         }
607         break;

609     case ACPI_DMT_GAS:

611         ByteLength = sizeof (ACPI_GENERIC_ADDRESS);
612         break;

614     case ACPI_DMT_HESTNTFY:

616         ByteLength = sizeof (ACPI_HEST_NOTIFY);
617         break;

619     case ACPI_DMT_BUFFER:

621         Value = DtGetFieldValue (Field);
622         if (Value)
623         {
624             ByteLength = DtGetBufferLength (Value);
625         }
626         else
627         {   /* At this point, this is a fatal error */

629             sprintf (MsgBuffer, "Expected \"%s\"", Info->Name);
630             DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, MsgBuffer);
631             return (0);
632         }
633         break;

635     case ACPI_DMT_BUF10:

637         ByteLength = 10;
638         break;

640     case ACPI_DMT_BUF16:
641     case ACPI_DMT_UUID:

643         ByteLength = 16;
644         break;

646     case ACPI_DMT_BUF128:

648         ByteLength = 128;
649         break;

651     case ACPI_DMT_UNICODE:

653         Value = DtGetFieldValue (Field);
```

```
655         /* TBD: error if Value is NULL? (as below?) */

657         ByteLength = (ACPI_STRLEN (Value) + 1) * sizeof(UINT16);
658         break;

660     default:

662         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field, "Invalid table opcode");
663         return (0);
664     }

666     return (ByteLength);
667 }


670 /******************************************************************************
671  *
672  * FUNCTION:    DtSum
673  *
674  * PARAMETERS:  DT_WALK_CALLBACK:
675  *              Subtable            - Subtable
676  *              Context             - Unused
677  *              ReturnValue         - Store the checksum of subtable
678  *
679  * RETURN:      Status
680  *
681  * DESCRIPTION: Get the checksum of subtable
682  *
683  *****************************************************************************/

685 static void
686 DtSum (
687     DT_SUBTABLE             *Subtable,
688     void                    *Context,
689     void                    *ReturnValue)
690 {
691     UINT8                   Checksum;
692     UINT8                   *Sum = ReturnValue;


695     Checksum = AcpiTbChecksum (Subtable->Buffer, Subtable->Length);
696     *Sum = (UINT8) (*Sum + Checksum);
697 }


700 /******************************************************************************
701  *
702  * FUNCTION:    DtSetTableChecksum
703  *
704  * PARAMETERS:  ChecksumPointer     - Where to return the checksum
705  *
706  * RETURN:      None
707  *
708  * DESCRIPTION: Set checksum of the whole data table into the checksum field
709  *
710  *****************************************************************************/

712 void
713 DtSetTableChecksum (
714     UINT8                   *ChecksumPointer)
715 {
716     UINT8                   Checksum = 0;
717     UINT8                   OldSum;
```



```
720     DtWalkTableTree (Gbl_RootTable, DtSum, NULL, &Checksum);

722     OldSum = *ChecksumPointer;
723     Checksum = (UINT8) (Checksum - OldSum);

725     /* Compute the final checksum */

727     Checksum = (UINT8) (0 - Checksum);
728     *ChecksumPointer = Checksum;
729 }


732 /******************************************************************************
733  *
734  * FUNCTION:    DtSetTableLength
735  *
736  * PARAMETERS:  None
737  *
738  * RETURN:      None
739  *
740  * DESCRIPTION: Walk the subtables and set all the length fields
741  *
742  *****************************************************************************/

744 void
745 DtSetTableLength (
746     void)
747 {
748     DT_SUBTABLE             *ParentTable;
749     DT_SUBTABLE             *ChildTable;


752     ParentTable = Gbl_RootTable;
753     ChildTable = NULL;

755     if (!ParentTable)
756     {
757         return;
758     }

760     DtSetSubtableLength (ParentTable);

762     while (1)
763     {
764         ChildTable = DtGetNextSubtable (ParentTable, ChildTable);
765         if (ChildTable)
766         {
767             if (ChildTable->LengthField)
768             {
769                 DtSetSubtableLength (ChildTable);
770             }

772             if (ChildTable->Child)
773             {
774                 ParentTable = ChildTable;
775                 ChildTable = NULL;
776             }
777             else
778             {
779                 ParentTable->TotalLength += ChildTable->TotalLength;
780                 if (ParentTable->LengthField)
781                 {
782                     DtSetSubtableLength (ParentTable);
783                 }
784             }
785         }
```

```
        else
        {
            ChildTable = ParentTable;

            if (ChildTable == Gbl_RootTable)
            {
                break;
            }

            ParentTable = DtGetParentSubtable (ParentTable);

            ParentTable->TotalLength += ChildTable->TotalLength;
            if (ParentTable->LengthField)
            {
                DtSetSubtableLength (ParentTable);
            }
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    DtWalkTableTree
 *
 * PARAMETERS:  StartTable          - Subtable in the tree where walking begins
 *              UserFunction        - Called during the walk
 *              Context             - Passed to user function
 *              ReturnValue         - The return value of UserFunction
 *
 * RETURN:      None
 *
 * DESCRIPTION: Performs a depth-first walk of the subtable tree
 *
 *****************************************************************************/

void
DtWalkTableTree (
    DT_SUBTABLE             *StartTable,
    DT_WALK_CALLBACK        UserFunction,
    void                    *Context,
    void                    *ReturnValue)
{
    DT_SUBTABLE             *ParentTable;
    DT_SUBTABLE             *ChildTable;


    ParentTable = StartTable;
    ChildTable = NULL;

    if (!ParentTable)
    {
        return;
    }

    UserFunction (ParentTable, Context, ReturnValue);

    while (1)
    {
        ChildTable = DtGetNextSubtable (ParentTable, ChildTable);
        if (ChildTable)
        {
            UserFunction (ChildTable, Context, ReturnValue);

            if (ChildTable->Child)
            {
```



```
                ParentTable = ChildTable;
                ChildTable = NULL;
            }
        }
        else
        {
            ChildTable = ParentTable;
            if (ChildTable == Gbl_RootTable)
            {
                break;
            }

            ParentTable = DtGetParentSubtable (ParentTable);

            if (ChildTable->Peer == StartTable)
            {
                break;
            }
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    DtFreeFieldList
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Free the field list
 *
 *****************************************************************************/

void
DtFreeFieldList (
    void)
{
    DT_FIELD                *Field = Gbl_FieldList;
    DT_FIELD                *NextField;


    /* Walk and free entire field list */

    while (Field)
    {
        NextField = Field->Next; /* Save link */

        if (!(Field->Flags & DT_FIELD_NOT_ALLOCATED))
        {
            ACPI_FREE (Field->Name);
            ACPI_FREE (Field->Value);
        }

        ACPI_FREE (Field);
        Field = NextField;
    }
}
```

```
**********************************************************
    3581 Thu Dec 26 13:48:39 2013
new/usr/src/common/acpica/compiler/new_table.txt
acpica-unix2-20130823
**********************************************************
```

How to add a new ACPI table to ACPICA and the iASL compiler.

There are four main tasks that are needed to provide support for a
new ACPI table:
    1) Create a full definition of the table and any subtables
       in the ACPICA headers.
    2) Add disassembler support for the new table
    3) Add iASL table compiler support for the new table
    4) Create a default template for the new table for iASL -T
       option.

Notes for each of these tasks provided below.

1) Header Support

New tables should be added to the appropriate header:
    actbl2.h: Used for new tables that are not defined in the ACPI spec.
    actbl3.h: Used for new tables that are defined in the ACPI spec.

Use ACPI_TABLE_HEADER for the common ACPI table header.
Subtables should be defined separately from the main table.
Don't add placeholder fields for subtables and other multiple data items.
    (Don't use xxxxx[1] for a field that can have multiple items.)
    The disassembler and data table compiler depends on this.
For tables not defined in the ACPI spec, add a comment to indicate where
    the table came from.
Use other table definitions for additional guidance.

2) iASL Disassembler Support

Add definition of the table (and subtables) in common/dmtbinfo.c
Add table access macro(s) of the form ACPI_xxxx_OFFSET
Add ACPI_DMT_TERMINATOR at the end of every table/subtable definition

Add externals for the table/subtable definitions in acdisasm.h
Add an entry for the new table in the AcpiDmTableData in common/dmtable.c

If there are no subtables, add the AcpiDmTableInfoXXXX name to the
    AcpiDmTableData and it will automatically be disassembled.

If there are subtables, a dump routine must be written:
Add an AcpiDmDumpXXXX function to dmtbdump.c -- note, code for another
    similar table can often be ported for the new table.
Add an external for this function to acdisasm.h
Add this function to the AcpiDmTableData entry for the new ACPI table

Debug/Test: Either find an existing example of the new ACPI table, or
    create one using the "generic ACPI table support" included in the
    iASL data table compiler. Use the -G option to force a
    generic compile. It is often best to create the table from scratch,
    since this clearly exposes the dependencies (lengths, offsets, etc.)
    that the Table Compiler support will need to generate.

3) iASL Table Compiler Support

Simple tables do not require a compile routine. The definition of the
    table in common/dmtbinfo.c (created in step 2 above) will suffice.

Complex tables with subtables will require a compile routine with a name
    of the form DtCompileXXXX.
Add a DtCompileXXXX function to the dttable.c module.
Add an external for this function in dtcompiler.h
Add this function to the AcpiDmTableData entry for the new ACPI table
    in common/dmtable.c

4) Template Support (-T iASL option)

Create an example of the new ACPI table. This example should create
    multiple subtables (if supported), and multiple instances of any
    variable length data.

Compile the example file with the -sc option. This will create a C
    array that contains the table contents.

Add this array to the dttemplate.h file. Name the array TemplateXXXX.
Add this array name to the AcpiDmTableData entry for the new ACPI table

Debug/Test: Create the template file. Compile the file. Disassemble the file.
    Compile the disassembly file.

```
**********************************************************
    7820 Thu Dec 26 13:48:39 2013
new/usr/src/common/acpica/compiler/preprocess.h
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: preprocess.h - header for iASL Preprocessor
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __PREPROCESS_H__

#ifndef _PREPROCESS
#define _PREPROCESS

#undef PR_EXTERN

#ifdef _DECLARE_PR_GLOBALS
#define PR_EXTERN
#define PR_INIT_GLOBAL(a,b)         (a)=(b)
#else
#define PR_EXTERN                   extern
#define PR_INIT_GLOBAL(a,b)         (a)
#endif


/*
 * Configuration
```



```
 */
#define PR_MAX_MACRO_ARGS       32              /* Max number of macro args */
#define PR_MAX_ARG_INSTANCES    24              /* Max instances of any one arg
#define PR_LINES_PER_BLOCK      4096            /* Max input source lines per bl


/*
 * Local defines and macros
 */
#define PR_TOKEN_SEPARATORS     " ,(){}\t\n"
#define PR_MACRO_SEPARATORS     " ,(){}~!*/%+-<>=&^|\"\t\n"
#define PR_MACRO_ARGUMENTS      " ,\t\n"
#define PR_EXPR_SEPARATORS      " ,(){}~!*/%+-<>=&^|\"\t\n"

#define PR_PREFIX_ID            "Pr(%.4u) - "           /* Used for debug outp

#define THIS_TOKEN_OFFSET(t)    ((t-Gbl_MainTokenBuffer) + 1)


/*
 * Preprocessor structures
 */
typedef struct pr_macro_arg
{
    char                        *Name;
    UINT32                      Offset[PR_MAX_ARG_INSTANCES];
    UINT16                      UseCount;

} PR_MACRO_ARG;

typedef struct pr_define_info
{
    struct pr_define_info       *Previous;
    struct pr_define_info       *Next;
    char                        *Identifier;
    char                        *Replacement;
    char                        *Body;          /* Macro body */
    PR_MACRO_ARG                *Args;          /* Macro arg list */
    UINT16                      ArgCount;       /* Macro arg count */
    BOOLEAN                     Persist;        /* Keep for entire compiler run

} PR_DEFINE_INFO;

typedef struct pr_directive_info
{
    char                        *Name;          /* Directive name */
    UINT8                       ArgCount;       /* Required # of args */

} PR_DIRECTIVE_INFO;

typedef struct pr_operator_info
{
    char                        *Op;

} PR_OPERATOR_INFO;

typedef struct pr_file_node
{
    struct pr_file_node         *Next;
    FILE                        *File;
    char                        *Filename;
    UINT32                      CurrentLineNumber;

} PR_FILE_NODE;

#define MAX_ARGUMENT_LENGTH     24
```

```
129 typedef struct directive_info
130 {
131     struct directive_info       *Next;
132     char                        Argument[MAX_ARGUMENT_LENGTH];
133     int                         Directive;
134     BOOLEAN                     IgnoringThisCodeBlock;

136 } DIRECTIVE_INFO;


139 /*
140  * Globals
141  */
142 #if 0 /* TBD for macros */
143 PR_EXTERN char                  PR_INIT_GLOBAL (*XXXEvalBuffer, NULL); /* [ASL_L
144 #endif

146 PR_EXTERN char                  PR_INIT_GLOBAL (*Gbl_MainTokenBuffer, NULL); /*
147 PR_EXTERN char                  PR_INIT_GLOBAL (*Gbl_MacroTokenBuffer, NULL); /*
148 PR_EXTERN char                  PR_INIT_GLOBAL (*Gbl_ExpressionTokenBuffer, NULL

150 PR_EXTERN UINT32                Gbl_PreprocessorLineNumber;
151 PR_EXTERN int                   Gbl_IfDepth;
152 PR_EXTERN PR_FILE_NODE          *Gbl_InputFileList;
153 PR_EXTERN PR_DEFINE_INFO        PR_INIT_GLOBAL (*Gbl_DefineList, NULL);
154 PR_EXTERN BOOLEAN               PR_INIT_GLOBAL (Gbl_PreprocessorError, FALSE);
155 PR_EXTERN BOOLEAN               PR_INIT_GLOBAL (Gbl_IgnoringThisCodeBlock, FALSE
156 PR_EXTERN DIRECTIVE_INFO        PR_INIT_GLOBAL (*Gbl_DirectiveStack, NULL);

158 /*
159  * prscan - Preprocessor entry
160  */
161 void
162 PrInitializePreprocessor (
163     void);

165 void
166 PrInitializeGlobals (
167     void);

169 void
170 PrTerminatePreprocessor (
171     void);

173 void
174 PrDoPreprocess (
175     void);

177 UINT64
178 PrIsDefined (
179     char                    *Identifier);

181 UINT64
182 PrResolveDefine (
183     char                    *Identifier);

185 int
186 PrInitLexer (
187     char                    *String);

189 void
190 PrTerminateLexer (
191     void);
```



```
194 /*
195  * prmacros - Support for #defines and macros
196  */
197 void
198 PrDumpPredefinedNames (
199     void);

201 PR_DEFINE_INFO *
202 PrAddDefine (
203     char                    *Token,
204     char                    *Token2,
205     BOOLEAN                 Persist);

207 void
208 PrRemoveDefine (
209     char                    *DefineName);

211 PR_DEFINE_INFO *
212 PrMatchDefine (
213     char                    *MatchString);

215 void
216 PrAddMacro (
217     char                    *Name,
218     char                    **Next);

220 void
221 PrDoMacroInvocation (
222     char                    *TokenBuffer,
223     char                    *MacroStart,
224     PR_DEFINE_INFO          *DefineInfo,
225     char                    **Next);


228 /*
229  * prexpress - #if expression support
230  */
231 ACPI_STATUS
232 PrResolveIntegerExpression (
233     char                    *Line,
234     UINT64                  *ReturnValue);

236 char *
237 PrPrioritizeExpression (
238     char                    *OriginalLine);

240 /*
241  * prparser - lex/yacc expression parser
242  */
243 UINT64
244 PrEvaluateExpression (
245     char                    *ExprString);


248 /*
249  * prutils - Preprocesor utilities
250  */
251 char *
252 PrGetNextToken (
253     char                    *Buffer,
254     char                    *MatchString,
255     char                    **Next);

257 void
258 PrError (
259     UINT8                   Level,
```

```
260     UINT8                   MessageId,
261     UINT32                  Column);

263 void
264 PrReplaceData (
265     char                    *Buffer,
266     UINT32                  LengthToRemove,
267     char                    *BufferToAdd,
268     UINT32                  LengthToAdd);

270 void
271 PrOpenIncludeFile (
272     char                    *Filename);

274 FILE *
275 PrOpenIncludeWithPrefix (
276     char                    *PrefixDir,
277     char                    *Filename);

279 void
280 PrPushInputFileStack (
281     FILE                    *InputFile,
282     char                    *Filename);

284 BOOLEAN
285 PrPopInputFileStack (
286     void);

288 #endif
```

```
**********************************************************
    9317 Thu Dec 26 13:48:39 2013
new/usr/src/common/acpica/compiler/prexpress.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: prexpress - Preprocessor #if expression support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"


#define _COMPONENT          ASL_PREPROCESSOR
        ACPI_MODULE_NAME    ("prexpress")

/* Local prototypes */

static char *
PrExpandMacros (
    char                    *Line);


#ifdef _UNDER_DEVELOPMENT
/******************************************************************************
 *
 * FUNCTION:    PrUnTokenize
```



```
 *
 * PARAMETERS:  Buffer              - Token Buffer
 *              Next                - "Next" buffer from GetNextToken
 *
 * RETURN:      None
 *
 * DESCRIPTION: Un-tokenized the current token buffer. The implementation is
 *              to simply set the null inserted by GetNextToken to a blank.
 *              If Next is NULL, there were no tokens found in the Buffer,
 *              so there is nothing to do.
 *
 *****************************************************************************/

static void
PrUnTokenize (
    char                    *Buffer,
    char                    *Next)
{
    UINT32                  Length = strlen (Buffer);


    if (!Next)
    {
        return;
    }
    if (Buffer[Length] != '\n')
    {
        Buffer[strlen(Buffer)] = ' ';
    }
}
#endif


/******************************************************************************
 *
 * FUNCTION:    PrExpandMacros
 *
 * PARAMETERS:  Line                - Pointer into the current line
 *
 * RETURN:      Updated pointer into the current line
 *
 * DESCRIPTION: Expand any macros found in the current line buffer.
 *
 *****************************************************************************/

static char *
PrExpandMacros (
    char                    *Line)
{
    char                    *Token;
    char                    *ReplaceString;
    PR_DEFINE_INFO          *DefineInfo;
    ACPI_SIZE               TokenOffset;
    char                    *Next;
    int                     OffsetAdjust;


    strcpy (Gbl_ExpressionTokenBuffer, Gbl_CurrentLineBuffer);
    Token = PrGetNextToken (Gbl_ExpressionTokenBuffer, PR_EXPR_SEPARATORS, &Next
    OffsetAdjust = 0;

    while (Token)
    {
        DefineInfo = PrMatchDefine (Token);
        if (DefineInfo)
        {
```

```
128             if (DefineInfo->Body)
129             {
130                 /* This is a macro. TBD: Is this allowed? */

132                 DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
133                     "Matched Macro: %s->%s\n",
134                     Gbl_CurrentLineNumber, DefineInfo->Identifier,
135                     DefineInfo->Replacement);

137                 PrDoMacroInvocation (Gbl_ExpressionTokenBuffer, Token,
138                     DefineInfo, &Next);
139             }
140             else
141             {
142                 ReplaceString = DefineInfo->Replacement;

144                 /* Replace the name in the original line buffer */

146                 TokenOffset = Token - Gbl_ExpressionTokenBuffer + OffsetAdjust;
147                 PrReplaceData (
148                     &Gbl_CurrentLineBuffer[TokenOffset], strlen (Token),
149                     ReplaceString, strlen (ReplaceString));

151                 /* Adjust for length difference between old and new name length

153                 OffsetAdjust += strlen (ReplaceString) - strlen (Token);

155                 DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
156                     "Matched #define within expression: %s->%s\n",
157                     Gbl_CurrentLineNumber, Token,
158                     *ReplaceString ? ReplaceString : "(NULL STRING)");
159             }
160         }

162         Token = PrGetNextToken (NULL, PR_EXPR_SEPARATORS, &Next);
163     }

165     return (Line);
166 }


169 /*******************************************************************************
170  *
171  * FUNCTION:    PrIsDefined
172  *
173  * PARAMETERS:  Identifier          - Name to be resolved
174  *
175  * RETURN:      64-bit boolean integer value
176  *
177  * DESCRIPTION: Returns TRUE if the name is defined, FALSE otherwise (0).
178  *
179  ******************************************************************************/

181 UINT64
182 PrIsDefined (
183     char                    *Identifier)
184 {
185     UINT64                  Value;
186     PR_DEFINE_INFO          *DefineInfo;


189     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
190         "**** Is defined?:  %s\n", Gbl_CurrentLineNumber, Identifier);

192     Value = 0; /* Default is "Not defined" -- FALSE */
```



```
194     DefineInfo = PrMatchDefine (Identifier);
195     if (DefineInfo)
196     {
197         Value = ACPI_UINT64_MAX; /* TRUE */
198     }

200     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
201         "[#if defined %s] resolved to: %8.8X%8.8X\n",
202         Gbl_CurrentLineNumber, Identifier, ACPI_FORMAT_UINT64 (Value));

204     return (Value);
205 }


208 /*******************************************************************************
209  *
210  * FUNCTION:    PrResolveDefine
211  *
212  * PARAMETERS:  Identifier          - Name to be resolved
213  *
214  * RETURN:      A 64-bit boolean integer value
215  *
216  * DESCRIPTION: Returns TRUE if the name is defined, FALSE otherwise (0).
217  *
218  ******************************************************************************/

220 UINT64
221 PrResolveDefine (
222     char                    *Identifier)
223 {
224     UINT64                  Value;
225     PR_DEFINE_INFO          *DefineInfo;


228     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
229         "**** Resolve #define:  %s\n", Gbl_CurrentLineNumber, Identifier);

231     Value = 0; /* Default is "Not defined" -- FALSE */

233     DefineInfo = PrMatchDefine (Identifier);
234     if (DefineInfo)
235     {
236         Value = ACPI_UINT64_MAX; /* TRUE */
237     }

239     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
240         "[#if defined %s] resolved to: %8.8X%8.8X\n",
241         Gbl_CurrentLineNumber, Identifier, ACPI_FORMAT_UINT64 (Value));

243     return (Value);
244 }


247 /*******************************************************************************
248  *
249  * FUNCTION:    PrResolveIntegerExpression
250  *
251  * PARAMETERS:  Line                - Pointer to integer expression
252  *              ReturnValue         - Where the resolved 64-bit integer is
253  *                                    returned.
254  *
255  * RETURN:      Status
256  *
257  * DESCRIPTION: Resolve an integer expression to a single value. Supports
258  *              both integer constants and labels.
259  *
```

```
 260  *****************************************************************************/

 262 ACPI_STATUS
 263 PrResolveIntegerExpression (
 264     char                    *Line,
 265     UINT64                  *ReturnValue)
 266 {
 267     UINT64                  Result;
 268     char                    *ExpandedLine;


 271     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
 272         "***** Resolve #if:  %s\n", Gbl_CurrentLineNumber, Line);

 274     /* Expand all macros within the expression first */

 276     ExpandedLine = PrExpandMacros (Line);

 278     /* Now we can evaluate the expression */

 280     Result = PrEvaluateExpression (ExpandedLine);
 281     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
 282         "***** Expression Resolved to: %8.8X%8.8X\n",
 283         Gbl_CurrentLineNumber, ACPI_FORMAT_UINT64 (Result));

 285     *ReturnValue = Result;
 286     return (AE_OK);

 288 #if 0
 289 InvalidExpression:

 291     ACPI_FREE (EvalBuffer);
 292     PrError (ASL_ERROR, ASL_MSG_INVALID_EXPRESSION, 0);
 293     return (AE_ERROR);


 296 NormalExit:

 298     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
 299         "***** Expression Resolved to: %8.8X%8.8X\n",
 300         Gbl_CurrentLineNumber, ACPI_FORMAT_UINT64 (Value1));

 302     *ReturnValue = Value1;
 303     return (AE_OK);
 304 #endif
 305 }
```

```
**********************************************************
   16160 Thu Dec 26 13:48:39 2013
new/usr/src/common/acpica/compiler/prmacros.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: prmacros - Preprocessor #define macro support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"


#define _COMPONENT          ASL_PREPROCESSOR
        ACPI_MODULE_NAME    ("prmacros")


/******************************************************************************
 *
 * FUNCTION:    PrDumpPredefinedNames
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the list of #defines. Used as the preprocessor starts, to
 *              display the names that were defined on the command line.
```



```
 *              Debug information only.
 *
 *****************************************************************************/

void
PrDumpPredefinedNames (
    void)
{
    PR_DEFINE_INFO          *DefineInfo;


    DefineInfo = Gbl_DefineList;
    while (DefineInfo)
    {
        DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
            "Predefined #define: %s->%s\n",
            0, DefineInfo->Identifier, DefineInfo->Replacement);

        DefineInfo = DefineInfo->Next;
    }
}


/******************************************************************************
 *
 * FUNCTION:    PrAddDefine
 *
 * PARAMETERS:  Identifier          - Name to be replaced
 *              Replacement         - Replacement for Identifier
 *              Persist             - Keep define across multiple compiles?
 *
 * RETURN:      A new define_info struct. NULL on error.
 *
 * DESCRIPTION: Add a new #define to the global list
 *
 *****************************************************************************/

PR_DEFINE_INFO *
PrAddDefine (
    char                    *Identifier,
    char                    *Replacement,
    BOOLEAN                 Persist)
{
    char                    *IdentifierString;
    char                    *ReplacementString;
    PR_DEFINE_INFO          *DefineInfo;


    if (!Replacement)
    {
        Replacement = "";
    }

    /* Check for already-defined first */

    DefineInfo = PrMatchDefine (Identifier);
    if (DefineInfo)
    {
        DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID,
            "#define: name already exists: %s\n",
            Gbl_CurrentLineNumber, Identifier);

        /*
         * Name already exists. This is only an error if the target name
         * is different.
         */
```

```
128         if (strcmp (Replacement, DefineInfo->Replacement))
129         {
130             PrError (ASL_ERROR, ASL_MSG_EXISTING_NAME,
131                 THIS_TOKEN_OFFSET (Identifier));
132
133             return (NULL);
134         }
135
136         return (DefineInfo);
137     }
138
139     /* Copy input strings */
140
141     IdentifierString = UtLocalCalloc (strlen (Identifier) + 1);
142     strcpy (IdentifierString, Identifier);
143
144     ReplacementString = UtLocalCalloc (strlen (Replacement) + 1);
145     strcpy (ReplacementString, Replacement);
146
147     /* Init and link new define info struct */
148
149     DefineInfo = UtLocalCalloc (sizeof (PR_DEFINE_INFO));
150     DefineInfo->Replacement = ReplacementString;
151     DefineInfo->Identifier = IdentifierString;
152     DefineInfo->Persist = Persist;
153
154     if (Gbl_DefineList)
155     {
156         Gbl_DefineList->Previous = DefineInfo;
157     }
158
159     DefineInfo->Next = Gbl_DefineList;
160     Gbl_DefineList = DefineInfo;
161     return (DefineInfo);
162 }
163
164
165 /*******************************************************************************
166  *
167  * FUNCTION:    PrRemoveDefine
168  *
169  * PARAMETERS:  DefineName          - Name of define to be removed
170  *
171  * RETURN:      None
172  *
173  * DESCRIPTION: Implements #undef. Remove a #define if found in the global
174  *              list. No error if the target of the #undef does not exist,
175  *              as per the C #undef definition.
176  *
177  ******************************************************************************/
178
179 void
180 PrRemoveDefine (
181     char                    *DefineName)
182 {
183     PR_DEFINE_INFO          *DefineInfo;
184
185
186     /* Match name and delete the node */
187
188     DefineInfo = Gbl_DefineList;
189     while (DefineInfo)
190     {
191         if (!strcmp (DefineName, DefineInfo->Identifier))
192         {
193             /* Remove from linked list */
```



```
194
195             if (DefineInfo->Previous)
196             {
197                 (DefineInfo->Previous)->Next = DefineInfo->Next;
198             }
199             else
200             {
201                 Gbl_DefineList = DefineInfo->Next;
202             }
203
204             if (DefineInfo->Next)
205             {
206                 (DefineInfo->Next)->Previous = DefineInfo->Previous;
207             }
208
209             free (DefineInfo);
210             return;
211         }
212
213         DefineInfo = DefineInfo->Next;
214     }
215
216     /*
217      * Name was not found. By definition of #undef, this is not
218      * an error, however.
219      */
220     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
221         "#undef: could not find %s\n",
222         Gbl_CurrentLineNumber, DefineName);
223 }
224
225
226 /*******************************************************************************
227  *
228  * FUNCTION:    PrMatchDefine
229  *
230  * PARAMETERS:  MatchString         - Name associated with the #define
231  *
232  * RETURN:      Matched string if found. NULL otherwise.
233  *
234  * DESCRIPTION: Find a name in global #define list
235  *
236  ******************************************************************************/
237
238 PR_DEFINE_INFO *
239 PrMatchDefine (
240     char                    *MatchString)
241 {
242     PR_DEFINE_INFO          *DefineInfo;
243
244
245     DefineInfo = Gbl_DefineList;
246     while (DefineInfo)
247     {
248         if (!strcmp (MatchString, DefineInfo->Identifier))
249         {
250             return (DefineInfo);
251         }
252
253         DefineInfo = DefineInfo->Next;
254     }
255
256     return (NULL);
257 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    PrAddMacro
 *
 * PARAMETERS:  Name                - Start of the macro definition
 *              Next                - "Next" buffer from GetNextToken
 *
 * RETURN:      None
 *
 * DESCRIPTION: Add a new macro to the list of #defines. Handles argument
 *              processing.
 *
 ******************************************************************************/

void
PrAddMacro (
    char                    *Name,
    char                    **Next)
{
    char                    *Token = NULL;
    ACPI_SIZE               TokenOffset;
    ACPI_SIZE               MacroBodyOffset;
    PR_DEFINE_INFO          *DefineInfo;
    PR_MACRO_ARG            *Args;
    char                    *Body;
    char                    *BodyInSource;
    UINT32                  i;
    UINT16                  UseCount = 0;
    UINT16                  ArgCount = 0;
    UINT32                  Depth = 1;
    UINT32                  EndOfArgList;
    char                    BufferChar;


    /* Find the end of the arguments list */

    TokenOffset = Name - Gbl_MainTokenBuffer + strlen (Name) + 1;
    while (1)
    {
        BufferChar = Gbl_CurrentLineBuffer[TokenOffset];
        if (BufferChar == '(')
        {
            Depth++;
        }
        else if (BufferChar == ')')
        {
            Depth--;
        }
        else if (BufferChar == 0)
        {
            PrError (ASL_ERROR, ASL_MSG_MACRO_SYNTAX, TokenOffset);
            return;
        }

        if (Depth == 0)
        {
            /* Found arg list end */

            EndOfArgList = TokenOffset;
            break;
        }

        TokenOffset++;
    }

    /* At this point, we know that we have a reasonable argument list */
```



```
    Args = UtLocalCalloc (sizeof (PR_MACRO_ARG) * PR_MAX_MACRO_ARGS);

    /* Get the macro argument names */

    for (i = 0; i < PR_MAX_MACRO_ARGS; i++)
    {
        Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
        if (!Token)
        {
            /* This is the case for a NULL macro body */

            BodyInSource = "";
            goto AddMacroToList;
        }

        /* Don't go beyond the argument list */

        TokenOffset = Token - Gbl_MainTokenBuffer + strlen (Token);
        if (TokenOffset > EndOfArgList)
        {
            break;
        }

        DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
            "Macro arg: %s \n",
            Gbl_CurrentLineNumber, Token);

        Args[i].Name = UtLocalCalloc (strlen (Token) + 1);
        strcpy (Args[i].Name, Token);

        Args[i].UseCount = 0;

        ArgCount++;
        if (ArgCount >= PR_MAX_MACRO_ARGS)
        {
            PrError (ASL_ERROR, ASL_MSG_TOO_MANY_ARGUMENTS, TokenOffset);
            return;
        }
    }

    /* Get the macro body. Token now points to start of body */

    MacroBodyOffset = Token - Gbl_MainTokenBuffer;

    /* Match each method arg in the macro body for later use */

    Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
    while (Token)
    {
        /* Search the macro arg list for matching arg */

        for (i = 0; Args[i].Name && (i < PR_MAX_MACRO_ARGS); i++)
        {
            /*
             * Save argument offset within macro body. This is the mechanism
             * used to expand the macro upon invocation.
             *
             * Handles multiple instances of the same argument
             */
            if (!strcmp (Token, Args[i].Name))
            {
                UseCount = Args[i].UseCount;

                Args[i].Offset[UseCount] = (Token - Gbl_MainTokenBuffer) - Macro
```

```
392                 DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
393                     "Macro Arg #%u: %s UseCount %u Offset %u \n",
394                     Gbl_CurrentLineNumber, i, Token,
395                     UseCount+1, Args[i].Offset[UseCount]);
396
397                 Args[i].UseCount++;
398                 if (Args[i].UseCount >= PR_MAX_ARG_INSTANCES)
399                 {
400                     PrError (ASL_ERROR, ASL_MSG_TOO_MANY_ARGUMENTS,
401                         THIS_TOKEN_OFFSET (Token));
402
403                     return;
404                 }
405                 break;
406             }
407         }
408
409         Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
410     }
411
412     BodyInSource = &Gbl_CurrentLineBuffer[MacroBodyOffset];
413
414
415 AddMacroToList:
416
417     /* Check if name is already defined first */
418
419     DefineInfo = PrMatchDefine (Name);
420     if (DefineInfo)
421     {
422         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
423             "#define: macro name already exists: %s\n",
424             Gbl_CurrentLineNumber, Name);
425
426         /* Error only if not exactly the same macro */
427
428         if (strcmp (DefineInfo->Body, BodyInSource) ||
429             (DefineInfo->ArgCount != ArgCount))
430         {
431             PrError (ASL_ERROR, ASL_MSG_EXISTING_NAME,
432                 THIS_TOKEN_OFFSET (Name));
433         }
434
435         return;
436     }
437
438     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
439         "Macro body: %s \n",
440         Gbl_CurrentLineNumber, BodyInSource);
441
442     /* Add macro to the #define list */
443
444     DefineInfo = PrAddDefine (Name, BodyInSource, FALSE);
445     if (DefineInfo)
446     {
447         Body = UtLocalCalloc (strlen (BodyInSource) + 1);
448         strcpy (Body, BodyInSource);
449
450         DefineInfo->Body = Body;
451         DefineInfo->Args = Args;
452         DefineInfo->ArgCount = ArgCount;
453     }
454 }
455
456
457 /******************************************************************************
```



```
458  *
459  * FUNCTION:    PrDoMacroInvocation
460  *
461  * PARAMETERS:  TokenBuffer         - Current line buffer
462  *              MacroStart          - Start of the macro invocation within
463  *                                    the token buffer
464  *              DefineInfo          - Info for this macro
465  *              Next                - "Next" buffer from GetNextToken
466  *
467  * RETURN:      None
468  *
469  * DESCRIPTION: Expand a macro invocation
470  *
471  *****************************************************************************/
472
473 void
474 PrDoMacroInvocation (
475     char                    *TokenBuffer,
476     char                    *MacroStart,
477     PR_DEFINE_INFO          *DefineInfo,
478     char                    **Next)
479 {
480     PR_MACRO_ARG            *Args;
481     char                    *Token = NULL;
482     UINT32                  TokenOffset;
483     UINT32                  Length;
484     UINT32                  i;
485
486
487     /* Take a copy of the macro body for expansion */
488
489     strcpy (Gbl_MacroTokenBuffer, DefineInfo->Body);
490
491     /* Replace each argument within the prototype body */
492
493     Args = DefineInfo->Args;
494     if (!Args->Name)
495     {
496         /* This macro has no arguments */
497
498         Token = PrGetNextToken (NULL, PR_MACRO_ARGUMENTS, Next);
499         if (!Token)
500         {
501             goto BadInvocation;
502         }
503
504         TokenOffset = (MacroStart - TokenBuffer);
505         Length = Token - MacroStart + strlen (Token) + 1;
506
507         PrReplaceData (
508             &Gbl_CurrentLineBuffer[TokenOffset], Length,
509             Gbl_MacroTokenBuffer, strlen (Gbl_MacroTokenBuffer));
510         return;
511     }
512
513     while (Args->Name)
514     {
515         /* Get the next argument from macro invocation */
516
517         Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
518         if (!Token)
519         {
520             goto BadInvocation;
521         }
522
523         /* Replace all instances of this argument */
```

```
525         for (i = 0; i < Args->UseCount; i++)
526         {
527             /* Offset zero indicates "arg not used" */
528             /* TBD: Not really needed now, with UseCount available */
529
530             if (Args->Offset[i] == 0)
531             {
532                 break;
533             }
534
535             PrReplaceData (
536                 &Gbl_MacroTokenBuffer[Args->Offset[i]], strlen (Args->Name),
537                 Token, strlen (Token));
538
539             DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
540                 "ExpandArg: %s \n",
541                 Gbl_CurrentLineNumber, Gbl_MacroTokenBuffer);
542         }
543
544         Args++;
545     }
546
547     /* TBD: need to make sure macro was not invoked with too many arguments */
548
549     if (!Token)
550     {
551         return;
552     }
553
554     /* Replace the entire macro invocation with the expanded macro */
555
556     TokenOffset = (MacroStart - TokenBuffer);
557     Length = Token - MacroStart + strlen (Token) + 1;
558
559     PrReplaceData (
560         &Gbl_CurrentLineBuffer[TokenOffset], Length,
561         Gbl_MacroTokenBuffer, strlen (Gbl_MacroTokenBuffer));
562
563     return;
564
565
566 BadInvocation:
567     PrError (ASL_ERROR, ASL_MSG_INVALID_INVOCATION,
568         THIS_TOKEN_OFFSET (MacroStart));
569
570     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
571         "Bad macro invocation: %s \n",
572         Gbl_CurrentLineNumber, Gbl_MacroTokenBuffer);
573     return;
574 }
```

```
**********************************************************
    4947 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/compiler/prparser.l
acpica-unix2-20130823
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: prparser.l - Flex input file for preprocessor lexer
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "prparser.y.h"

/* Buffer to pass strings to the parser */

#define STRING_SETUP    strcpy (StringBuffer, PrParsertext);\
    PrParserlval.str = StringBuffer

#define YY_NO_INPUT     /* No file input, we use strings only */

#define _COMPONENT          ACPI_COMPILER
        ACPI_MODULE_NAME    ("prscanner")
%}

%option noyywrap
%option nounput
```



```
Number          [0-9a-fA-F]+
HexNumber       0[xX][0-9a-fA-F]+
WhiteSpace      [ \t\v\r]+
NewLine         [\n]
Identifier      [a-zA-Z][0-9a-zA-Z]*

%%

\(              return (EXPOP_PAREN_OPEN);
\)              return (EXPOP_PAREN_CLOSE);
\~              return (EXPOP_ONES_COMPLIMENT);
\!              return (EXPOP_LOGICAL_NOT);
\*              return (EXPOP_MULTIPLY);
\/              return (EXPOP_DIVIDE);
\%              return (EXPOP_MODULO);
\+              return (EXPOP_ADD);
\-              return (EXPOP_SUBTRACT);
">>"            return (EXPOP_SHIFT_RIGHT);
"<<"            return (EXPOP_SHIFT_LEFT);
\<              return (EXPOP_LESS);
\>              return (EXPOP_GREATER);
"<="            return (EXPOP_LESS_EQUAL);
">="            return (EXPOP_GREATER_EQUAL);
"=="            return (EXPOP_EQUAL);
"!="            return (EXPOP_NOT_EQUAL);
\&              return (EXPOP_AND);
\^              return (EXPOP_XOR);
\|              return (EXPOP_OR);
"&&"            return (EXPOP_LOGICAL_AND);
"||"            return (EXPOP_LOGICAL_OR);

"defined"       return (EXPOP_DEFINE);
{Identifier}    {STRING_SETUP; return (EXPOP_IDENTIFIER);}

<<EOF>>         return (EXPOP_EOF); /* null end-of-string */

{Number}        return (EXPOP_NUMBER);
{HexNumber}     return (EXPOP_HEX_NUMBER);
{NewLine}       return (EXPOP_NEW_LINE);
{WhiteSpace}    /* Ignore */

.               return (EXPOP_EOF);
%%

/*
 * Local support functions
 */
YY_BUFFER_STATE         LexBuffer;


/*******************************************************************************
 *
 * FUNCTION:    PrInitLexer
 *
 * PARAMETERS:  String              - Input string to be parsed
 *
 * RETURN:      TRUE if parser returns NULL. FALSE otherwise.
 *
 * DESCRIPTION: Initialization routine for lexer. The lexer needs
 *              a buffer to handle strings instead of a file.
 *
 ******************************************************************************/

int
PrInitLexer (
    char                    *String)
```

```
128 {
130     LexBuffer = yy_scan_string (String);
131     return (LexBuffer == NULL);
132 }
135 /*******************************************************************************
136  *
137  * FUNCTION:    PrTerminateLexer
138  *
139  * PARAMETERS:  None
140  *
141  * RETURN:      None
142  *
143  * DESCRIPTION: Termination routine for thelexer.
144  *
145  ******************************************************************************/
147 void
148 PrTerminateLexer (
149     void)
150 {
152     yy_delete_buffer (LexBuffer);
153 }
```

```
**********************************************************
   10174 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/compiler/prparser.y
acpica-unix2-20130823
**********************************************************
```

```
%{
/******************************************************************************
 *
 * Module Name: prparser.y - Bison input file for preprocessor parser
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"

#define _COMPONENT          ASL_PREPROCESSOR
        ACPI_MODULE_NAME    ("prparser")

int                         PrParserlex (void);
int                         PrParserparse (void);
void                        PrParsererror (char const *msg);
extern char                 *PrParsertext;

UINT64                      PrParserResult; /* Expression return value */

/* Bison/yacc configuration */

#define yytname             PrParsername
#define YYDEBUG             1               /* Enable debug output */
```



```
#define YYERROR_VERBOSE     1               /* Verbose error messages */
#define YYFLAG              -32768

/* Define YYMALLOC/YYFREE to prevent redefinition errors  */

#define YYMALLOC            malloc
#define YYFREE              free
%}

%union
{
     UINT64                 value;
     UINT32                 op;
     char                   *str;
}

/*! [Begin] no source code translation */

%type  <value>  Expression

%token <op>     EXPOP_EOF
%token <op>     EXPOP_NEW_LINE
%token <op>     EXPOP_NUMBER
%token <op>     EXPOP_HEX_NUMBER
%token <op>     EXPOP_RESERVED1
%token <op>     EXPOP_RESERVED2
%token <op>     EXPOP_PAREN_OPEN
%token <op>     EXPOP_PAREN_CLOSE

%left <op>      EXPOP_LOGICAL_OR
%left <op>      EXPOP_LOGICAL_AND
%left <op>      EXPOP_OR
%left <op>      EXPOP_XOR
%left <op>      EXPOP_AND
%left <op>      EXPOP_EQUAL EXPOP_NOT_EQUAL
%left <op>      EXPOP_GREATER EXPOP_LESS EXPOP_GREATER_EQUAL EXPOP_LESS_EQUAL
%left <op>      EXPOP_SHIFT_RIGHT EXPOP_SHIFT_LEFT
%left <op>      EXPOP_ADD EXPOP_SUBTRACT
%left <op>      EXPOP_MULTIPLY EXPOP_DIVIDE EXPOP_MODULO
%right <op>     EXPOP_ONES_COMPLIMENT EXPOP_LOGICAL_NOT

/* Tokens above must be kept in synch with dtparser.y */

%token <op>     EXPOP_DEFINE
%token <op>     EXPOP_IDENTIFIER

%%

/*
 *  Operator precedence rules (from K&R)
 *
 *  1)      ( )
 *  2)      ! ~ (unary operators that are supported here)
 *  3)      *   /   %
 *  4)      +   -
 *  5)      >>  <<
 *  6)      <   >   <=  >=
 *  7)      ==  !=
 *  8)      &
 *  9)      ^
 *  10)     |
 *  11)     &&
 *  12)     ||
 */

/*! [End] no source code translation !*/
```

```
129 Value
130     : Expression EXPOP_NEW_LINE                     { PrParserResult=$1; return
131     | Expression EXPOP_EOF                          { PrParserResult=$1; return
132     ;

134 Expression

136       /* Unary operators */

138     : EXPOP_LOGICAL_NOT         Expression          { $$ = DtDoOperator ($2, EXP
139     | EXPOP_ONES_COMPLIMENT     Expression          { $$ = DtDoOperator ($2, EXP

141       /* Binary operators */

143     | Expression EXPOP_MULTIPLY         Expression  { $$ = DtDoOperator ($1, EXP
144     | Expression EXPOP_DIVIDE           Expression  { $$ = DtDoOperator ($1, EXP
145     | Expression EXPOP_MODULO           Expression  { $$ = DtDoOperator ($1, EXP
146     | Expression EXPOP_ADD              Expression  { $$ = DtDoOperator ($1, EXP
147     | Expression EXPOP_SUBTRACT         Expression  { $$ = DtDoOperator ($1, EXP
148     | Expression EXPOP_SHIFT_RIGHT      Expression  { $$ = DtDoOperator ($1, EXP
149     | Expression EXPOP_SHIFT_LEFT       Expression  { $$ = DtDoOperator ($1, EXP
150     | Expression EXPOP_GREATER          Expression  { $$ = DtDoOperator ($1, EXP
151     | Expression EXPOP_LESS             Expression  { $$ = DtDoOperator ($1, EXP
152     | Expression EXPOP_GREATER_EQUAL    Expression  { $$ = DtDoOperator ($1, EXP
153     | Expression EXPOP_LESS_EQUAL       Expression  { $$ = DtDoOperator ($1, EXP
154     | Expression EXPOP_EQUAL            Expression  { $$ = DtDoOperator ($1, EXP
155     | Expression EXPOP_NOT_EQUAL        Expression  { $$ = DtDoOperator ($1, EXP
156     | Expression EXPOP_AND              Expression  { $$ = DtDoOperator ($1, EXP
157     | Expression EXPOP_XOR              Expression  { $$ = DtDoOperator ($1, EXP
158     | Expression EXPOP_OR               Expression  { $$ = DtDoOperator ($1, EXP
159     | Expression EXPOP_LOGICAL_AND      Expression  { $$ = DtDoOperator ($1, EXP
160     | Expression EXPOP_LOGICAL_OR       Expression  { $$ = DtDoOperator ($1, EXP

162       /* Parentheses: '(' Expression ')' */

164     | EXPOP_PAREN_OPEN          Expression
165         EXPOP_PAREN_CLOSE                           { $$ = $2;}

167       /* #if defined (ID) or #if defined ID */

169     | EXPOP_DEFINE EXPOP_PAREN_OPEN EXPOP_IDENTIFIER
170         EXPOP_PAREN_CLOSE                           { $$ = PrIsDefined (PrParser

172     | EXPOP_DEFINE EXPOP_IDENTIFIER                 { $$ = PrIsDefined (PrParser

174     | EXPOP_IDENTIFIER                              { $$ = PrResolveDefine (PrPa

176       /* Default base for a non-prefixed integer is 10 */

178     | EXPOP_NUMBER                                  { UtStrtoul64 (PrParsertext,

180       /* Standard hex number (0x1234) */

182     | EXPOP_HEX_NUMBER                              { UtStrtoul64 (PrParsertext,
183     ;
184 %%

186 /*
187  * Local support functions, including parser entry point
188  */
189 #define PR_FIRST_PARSE_OPCODE   EXPOP_EOF
190 #define PR_YYTNAME_START        3


193 /*****************************************************************************
```



```
194  *
195  * FUNCTION:    PrParsererror
196  *
197  * PARAMETERS:  Message             - Parser-generated error message
198  *
199  * RETURN:      None
200  *
201  * DESCRIPTION: Handler for parser errors
202  *
203  *****************************************************************************/

205 void
206 PrParsererror (
207     char const              *Message)
208 {
209     DtError (ASL_ERROR, ASL_MSG_SYNTAX,
210         NULL, (char *) Message);
211 }


214 /*****************************************************************************
215  *
216  * FUNCTION:    PrGetOpName
217  *
218  * PARAMETERS:  ParseOpcode         - Parser token (EXPOP_*)
219  *
220  * RETURN:      Pointer to the opcode name
221  *
222  * DESCRIPTION: Get the ascii name of the parse opcode for debug output
223  *
224  *****************************************************************************/

226 char *
227 PrGetOpName (
228     UINT32                  ParseOpcode)
229 {
230 #ifdef ASL_YYTNAME_START
231     /*
232      * First entries (PR_YYTNAME_START) in yytname are special reserved names.
233      * Ignore first 6 characters of name (EXPOP_)
234      */
235     return ((char *) yytname
236         [(ParseOpcode - PR_FIRST_PARSE_OPCODE) + PR_YYTNAME_START] + 6);
237 #else
238     return ("[Unknown parser generator]");
239 #endif
240 }


243 /*****************************************************************************
244  *
245  * FUNCTION:    PrEvaluateExpression
246  *
247  * PARAMETERS:  ExprString          - Expression to be evaluated. Must be
248  *                                    terminated by either a newline or a NUL
249  *                                    string terminator
250  *
251  * RETURN:      64-bit value for the expression
252  *
253  * DESCRIPTION: Main entry point for the DT expression parser
254  *
255  *****************************************************************************/

257 UINT64
258 PrEvaluateExpression (
259     char                    *ExprString)
```

```
260 {
262     DbgPrint (ASL_DEBUG_OUTPUT,
263         "**** Input expression: %s\n", ExprString);
265     /* Point lexer to the input string */
267     if (PrInitLexer (ExprString))
268     {
269         DtError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL,
270             NULL, "Could not initialize lexer");
271         return (0);
272     }
274     /* Parse/Evaluate the input string (value returned in PrParserResult) */
276     PrParserparse ();
277     PrTerminateLexer ();
279     DbgPrint (ASL_DEBUG_OUTPUT,
280         "**** Parser returned value: %u (%8.8X%8.8X)\n",
281         (UINT32) PrParserResult, ACPI_FORMAT_UINT64 (PrParserResult));
283     return (PrParserResult);
284 }
```

```
**********************************************************
   26485 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/compiler/prscan.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: prscan - Preprocessor start-up and file scan module
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define _DECLARE_PR_GLOBALS

#include "aslcompiler.h"
#include "dtcompiler.h"

/*
 * TBDs:
 *
 * No nested macros, maybe never
 * Implement ASL "Include" as well as "#include" here?
 */
#define _COMPONENT          ASL_PREPROCESSOR
        ACPI_MODULE_NAME    ("prscan")


/* Local prototypes */

static void
```



```
PrPreprocessInputFile (
    void);

static void
PrDoDirective (
    char                    *DirectiveToken,
    char                    **Next);

static int
PrMatchDirective (
    char                    *Directive);

static void
PrPushDirective (
    int                     Directive,
    char                    *Argument);

static ACPI_STATUS
PrPopDirective (
    void);

static void
PrDbgPrint (
    char                    *Action,
    char                    *DirectiveName);


/*
 * Supported preprocessor directives
 */
static const PR_DIRECTIVE_INFO      Gbl_DirectiveInfo[] =
{
    {"define",  1},
    {"elif",    0}, /* Converted to #else..#if internally */
    {"else",    0},
    {"endif",   0},
    {"error",   1},
    {"if",      1},
    {"ifdef",   1},
    {"ifndef",  1},
    {"include", 0}, /* Argument is not standard format, so 0 */
    {"line",    1},
    {"pragma",  1},
    {"undef",   1},
    {"warning", 1},
    {NULL,      0}
};

enum Gbl_DirectiveIndexes
{
    PR_DIRECTIVE_DEFINE = 0,
    PR_DIRECTIVE_ELIF,
    PR_DIRECTIVE_ELSE,
    PR_DIRECTIVE_ENDIF,
    PR_DIRECTIVE_ERROR,
    PR_DIRECTIVE_IF,
    PR_DIRECTIVE_IFDEF,
    PR_DIRECTIVE_IFNDEF,
    PR_DIRECTIVE_INCLUDE,
    PR_DIRECTIVE_LINE,
    PR_DIRECTIVE_PRAGMA,
    PR_DIRECTIVE_UNDEF,
    PR_DIRECTIVE_WARNING,
};

#define ASL_DIRECTIVE_NOT_FOUND     -1
```

```
130 /******************************************************************************
131  *
132  * FUNCTION:    PrInitializePreprocessor
133  *
134  * PARAMETERS:  None
135  *
136  * RETURN:      None
137  *
138  * DESCRIPTION: Startup initialization for the Preprocessor.
139  *
140  *****************************************************************************/

142 void
143 PrInitializePreprocessor (
144     void)
145 {
146     /* Init globals and the list of #defines */

148     PrInitializeGlobals ();
149     Gbl_DefineList = NULL;
150 }


153 /******************************************************************************
154  *
155  * FUNCTION:    PrInitializeGlobals
156  *
157  * PARAMETERS:  None
158  *
159  * RETURN:      None
160  *
161  * DESCRIPTION: Initialize globals for the Preprocessor. Used for startuup
162  *              initialization and re-initialization between compiles during
163  *              a multiple source file compile.
164  *
165  *****************************************************************************/

167 void
168 PrInitializeGlobals (
169     void)
170 {
171     /* Init globals */

173     Gbl_InputFileList = NULL;
174     Gbl_CurrentLineNumber = 0;
175     Gbl_PreprocessorLineNumber = 1;
176     Gbl_PreprocessorError = FALSE;

178     /* These are used to track #if/#else blocks (possibly nested) */

180     Gbl_IfDepth = 0;
181     Gbl_IgnoringThisCodeBlock = FALSE;
182     Gbl_DirectiveStack = NULL;
183 }


186 /******************************************************************************
187  *
188  * FUNCTION:    PrTerminatePreprocessor
189  *
190  * PARAMETERS:  None
191  *
192  * RETURN:      None
193  *
```



```
194  * DESCRIPTION: Termination of the preprocessor. Delete lists. Keep any
195  *              defines that were specified on the command line, in order to
196  *              support multiple compiles with a single compiler invocation.
197  *
198  *****************************************************************************/

200 void
201 PrTerminatePreprocessor (
202     void)
203 {
204     PR_DEFINE_INFO          *DefineInfo;


207     /*
208      * The persistent defines (created on the command line) are always at the
209      * end of the list. We save them.
210      */
211     while ((Gbl_DefineList) && (!Gbl_DefineList->Persist))
212     {
213         DefineInfo = Gbl_DefineList;
214         Gbl_DefineList = DefineInfo->Next;

216         ACPI_FREE (DefineInfo->Replacement);
217         ACPI_FREE (DefineInfo->Identifier);
218         ACPI_FREE (DefineInfo);
219     }
220 }


223 /******************************************************************************
224  *
225  * FUNCTION:    PrDoPreprocess
226  *
227  * PARAMETERS:  None
228  *
229  * RETURN:      None
230  *
231  * DESCRIPTION: Main entry point for the iASL Preprocessor. Input file must
232  *              be already open. Handles multiple input files via the
233  *              #include directive.
234  *
235  *****************************************************************************/

237 void
238 PrDoPreprocess (
239     void)
240 {
241     BOOLEAN                 MoreInputFiles;


244     DbgPrint (ASL_DEBUG_OUTPUT, "Starting preprocessing phase\n\n");


247     FlSeekFile (ASL_FILE_INPUT, 0);
248     PrDumpPredefinedNames ();

250     /* Main preprocessor loop, handles include files */

252     do
253     {
254         PrPreprocessInputFile ();
255         MoreInputFiles = PrPopInputFileStack ();

257     } while (MoreInputFiles);

259     /* Point compiler input to the new preprocessor output file (.i) */
```

```
261     FlCloseFile (ASL_FILE_INPUT);
262     Gbl_Files[ASL_FILE_INPUT].Handle = Gbl_Files[ASL_FILE_PREPROCESSOR].Handle;
263     AslCompilerin = Gbl_Files[ASL_FILE_INPUT].Handle;

265     /* Reset globals to allow compiler to run */

267     FlSeekFile (ASL_FILE_INPUT, 0);
268     Gbl_CurrentLineNumber = 1;

270     DbgPrint (ASL_DEBUG_OUTPUT, "Preprocessing phase complete \n\n");
271 }


274 /******************************************************************************
275  *
276  * FUNCTION:    PrPreprocessInputFile
277  *
278  * PARAMETERS:  None
279  *
280  * RETURN:      None
281  *
282  * DESCRIPTION: Preprocess one entire file, line-by-line.
283  *
284  * Input:  Raw user ASL from ASL_FILE_INPUT
285  * Output: Preprocessed file written to ASL_FILE_PREPROCESSOR
286  *
287  *****************************************************************************/

289 static void
290 PrPreprocessInputFile (
291     void)
292 {
293     UINT32                  Offset;
294     char                    *Token;
295     char                    *ReplaceString;
296     PR_DEFINE_INFO          *DefineInfo;
297     ACPI_SIZE               TokenOffset;
298     char                    *Next;
299     int                     OffsetAdjust;


302     /* Scan line-by-line. Comments and blank lines are skipped by this function

304     while ((Offset = DtGetNextLine (Gbl_Files[ASL_FILE_INPUT].Handle)) != ASL_EO
305     {
306         /* Need a copy of the input line for strok() */

308         strcpy (Gbl_MainTokenBuffer, Gbl_CurrentLineBuffer);
309         Token = PrGetNextToken (Gbl_MainTokenBuffer, PR_TOKEN_SEPARATORS, &Next)
310         OffsetAdjust = 0;

312         /* All preprocessor directives must begin with '#' */

314         if (Token && (*Token == '#'))
315         {
316             if (strlen (Token) == 1)
317             {
318                 Token = PrGetNextToken (NULL, PR_TOKEN_SEPARATORS, &Next);
319             }
320             else
321             {
322                 Token++;    /* Skip leading # */
323             }

325             /* Execute the directive, do not write line to output file */
```



```
327             PrDoDirective (Token, &Next);
328             continue;
329         }

331         /*
332          * If we are currently within the part of an IF/ELSE block that is
333          * FALSE, ignore the line and do not write it to the output file.
334          * This continues until an #else or #endif is encountered.
335          */
336         if (Gbl_IgnoringThisCodeBlock)
337         {
338             continue;
339         }

341         /* Match and replace all #defined names within this source line */

343         while (Token)
344         {
345             DefineInfo = PrMatchDefine (Token);
346             if (DefineInfo)
347             {
348                 if (DefineInfo->Body)
349                 {
350                     /* This is a macro */

352                     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
353                         "Matched Macro: %s->%s\n",
354                         Gbl_CurrentLineNumber, DefineInfo->Identifier,
355                         DefineInfo->Replacement);

357                     PrDoMacroInvocation (Gbl_MainTokenBuffer, Token,
358                         DefineInfo, &Next);
359                 }
360                 else
361                 {
362                     ReplaceString = DefineInfo->Replacement;

364                     /* Replace the name in the original line buffer */

366                     TokenOffset = Token - Gbl_MainTokenBuffer + OffsetAdjust;
367                     PrReplaceData (
368                         &Gbl_CurrentLineBuffer[TokenOffset], strlen (Token),
369                         ReplaceString, strlen (ReplaceString));

371                     /* Adjust for length difference between old and new name len

373                     OffsetAdjust += strlen (ReplaceString) - strlen (Token);

375                     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
376                         "Matched #define: %s->%s\n",
377                         Gbl_CurrentLineNumber, Token,
378                         *ReplaceString ? ReplaceString : "(NULL STRING)");
379                 }
380             }

382             Token = PrGetNextToken (NULL, PR_TOKEN_SEPARATORS, &Next);
383         }

385 #if 0
386 /* Line prefix */
387         FlPrintFile (ASL_FILE_PREPROCESSOR, "/* %14s  %.5u  i:%.5u */ ",
388             Gbl_Files[ASL_FILE_INPUT].Filename,
389             Gbl_CurrentLineNumber, Gbl_PreprocessorLineNumber);
390 #endif
```

```
392         /*
393          * Emit a #line directive if necessary, to keep the line numbers in
394          * the (.i) file synchronized with the original source code file, so
395          * that the correct line number appears in any error messages
396          * generated by the actual compiler.
397          */
398         if (Gbl_CurrentLineNumber > (Gbl_PreviousLineNumber + 1))
399         {
400             FlPrintFile (ASL_FILE_PREPROCESSOR, "#line %u\n",
401                 Gbl_CurrentLineNumber);
402         }

404         Gbl_PreviousLineNumber = Gbl_CurrentLineNumber;
405         Gbl_PreprocessorLineNumber++;

407         /*
408          * Now we can write the possibly modified source line to the
409          * preprocessor (.i) file
410          */
411         FlWriteFile (ASL_FILE_PREPROCESSOR, Gbl_CurrentLineBuffer,
412             strlen (Gbl_CurrentLineBuffer));
413     }
414 }


417 /******************************************************************************
418  *
419  * FUNCTION:    PrDoDirective
420  *
421  * PARAMETERS:  Directive               - Pointer to directive name token
422  *              Next                    - "Next" buffer from GetNextToken
423  *
424  * RETURN:      None.
425  *
426  * DESCRIPTION: Main processing for all preprocessor directives
427  *
428  *****************************************************************************/

430 static void
431 PrDoDirective (
432     char                    *DirectiveToken,
433     char                    **Next)
434 {
435     char                    *Token = Gbl_MainTokenBuffer;
436     char                    *Token2;
437     char                    *End;
438     UINT64                  Value;
439     ACPI_SIZE               TokenOffset;
440     int                     Directive;
441     ACPI_STATUS             Status;


444     if (!DirectiveToken)
445     {
446         goto SyntaxError;
447     }

449     Directive = PrMatchDirective (DirectiveToken);
450     if (Directive == ASL_DIRECTIVE_NOT_FOUND)
451     {
452         PrError (ASL_ERROR, ASL_MSG_UNKNOWN_DIRECTIVE,
453             THIS_TOKEN_OFFSET (DirectiveToken));

455         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
456             "#%s: Unknown directive\n",
457             Gbl_CurrentLineNumber, DirectiveToken);
```



```
458         return;
459     }

461     /*
462      * If we are currently ignoring this block and we encounter a #else or
463      * #elif, we must ignore their blocks also if the parent block is also
464      * being ignored.
465      */
466     if (Gbl_IgnoringThisCodeBlock)
467     {
468         switch (Directive)
469         {
470         case PR_DIRECTIVE_ELSE:
471         case PR_DIRECTIVE_ELIF:

473             if (Gbl_DirectiveStack && Gbl_DirectiveStack->IgnoringThisCodeBlock)
474             {
475                 PrDbgPrint ("Ignoring", Gbl_DirectiveInfo[Directive].Name);
476                 return;
477             }
478             break;

480         default:
481             break;
482         }
483     }

485     /*
486      * Need to always check for #else, #elif, #endif regardless of
487      * whether we are ignoring the current code block, since these
488      * are conditional code block terminators.
489      */
490     switch (Directive)
491     {
492     case PR_DIRECTIVE_ELSE:

494         Gbl_IgnoringThisCodeBlock = !(Gbl_IgnoringThisCodeBlock);
495         PrDbgPrint ("Executing", "else block");
496         return;

498     case PR_DIRECTIVE_ELIF:

500         Gbl_IgnoringThisCodeBlock = !(Gbl_IgnoringThisCodeBlock);
501         Directive = PR_DIRECTIVE_IF;

503         if (Gbl_IgnoringThisCodeBlock == TRUE)
504         {
505             /* Not executing the ELSE part -- all done here */
506             PrDbgPrint ("Ignoring", "elif block");
507             return;
508         }

510         /*
511          * After this, we will execute the IF part further below.
512          * First, however, pop off the original #if directive.
513          */
514         if (ACPI_FAILURE (PrPopDirective ()))
515         {
516             PrError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL,
517                 THIS_TOKEN_OFFSET (DirectiveToken));
518         }

520         PrDbgPrint ("Executing", "elif block");
521         break;

523     case PR_DIRECTIVE_ENDIF:
```

```
525         PrDbgPrint ("Executing", "endif");

527         /* Pop the owning #if/#ifdef/#ifndef */

529         if (ACPI_FAILURE (PrPopDirective ()))
530         {
531             PrError (ASL_ERROR, ASL_MSG_ENDIF_MISMATCH,
532                 THIS_TOKEN_OFFSET (DirectiveToken));
533         }
534         return;

536     default:
537         break;
538     }

540     /* Most directives have at least one argument */

542     if (Gbl_DirectiveInfo[Directive].ArgCount == 1)
543     {
544         Token = PrGetNextToken (NULL, PR_TOKEN_SEPARATORS, Next);
545         if (!Token)
546         {
547             goto SyntaxError;
548         }
549     }

551     /*
552      * At this point, if we are ignoring the current code block,
553      * do not process any more directives (i.e., ignore them also.)
554      * For "if" style directives, open/push a new block anyway. We
555      * must do this to keep track of #endif directives
556      */
557     if (Gbl_IgnoringThisCodeBlock)
558     {
559         switch (Directive)
560         {
561         case PR_DIRECTIVE_IF:
562         case PR_DIRECTIVE_IFDEF:
563         case PR_DIRECTIVE_IFNDEF:

565             PrPushDirective (Directive, Token);
566             PrDbgPrint ("Ignoring", Gbl_DirectiveInfo[Directive].Name);
567             break;

569         default:
570             break;
571         }

573         return;
574     }

576     /*
577      * Execute the directive
578      */
579     PrDbgPrint ("Begin execution", Gbl_DirectiveInfo[Directive].Name);

581     switch (Directive)
582     {
583     case PR_DIRECTIVE_IF:

585         TokenOffset = Token - Gbl_MainTokenBuffer;

587         /* Need to expand #define macros in the expression string first */

589         Status = PrResolveIntegerExpression (
```



```
590             &Gbl_CurrentLineBuffer[TokenOffset-1], &Value);
591         if (ACPI_FAILURE (Status))
592         {
593             return;
594         }

596         PrPushDirective (Directive, Token);
597         if (!Value)
598         {
599             Gbl_IgnoringThisCodeBlock = TRUE;
600         }

602         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
603             "Resolved #if: %8.8X%8.8X %s\n",
604             Gbl_CurrentLineNumber, ACPI_FORMAT_UINT64 (Value),
605             Gbl_IgnoringThisCodeBlock ? "<Skipping Block>" : "<Executing Block>"
606         break;

608     case PR_DIRECTIVE_IFDEF:

610         PrPushDirective (Directive, Token);
611         if (!PrMatchDefine (Token))
612         {
613             Gbl_IgnoringThisCodeBlock = TRUE;
614         }

616         PrDbgPrint ("Evaluated", "ifdef");
617         break;

619     case PR_DIRECTIVE_IFNDEF:

621         PrPushDirective (Directive, Token);
622         if (PrMatchDefine (Token))
623         {
624             Gbl_IgnoringThisCodeBlock = TRUE;
625         }

627         PrDbgPrint ("Evaluated", "ifndef");
628         break;

630     case PR_DIRECTIVE_DEFINE:
631         /*
632          * By definition, if first char after the name is a paren,
633          * this is a function macro.
634          */
635         TokenOffset = Token - Gbl_MainTokenBuffer + strlen (Token);
636         if (*(&Gbl_CurrentLineBuffer[TokenOffset]) == '(')
637         {
638 #ifndef MACROS_SUPPORTED
639             AcpiOsPrintf ("%s ERROR - line %u: #define macros are not supported
640                 Gbl_CurrentLineBuffer, Gbl_CurrentLineNumber);
641             exit(1);
642 #else
643             PrAddMacro (Token, Next);
644 #endif
645         }
646         else
647         {
648             /* Use the remainder of the line for the #define */

650             Token2 = *Next;
651             if (Token2)
652             {
653                 while ((*Token2 == ' ') || (*Token2 == '\t'))
654                 {
655                     Token2++;
```

```
656                 }
657                 End = Token2;
658                 while (*End != '\n')
659                 {
660                     End++;
661                 }
662                 *End = 0;
663             }
664             else
665             {
666                 Token2 = "";
667             }
668 #if 0
669             Token2 = PrGetNextToken (NULL, "\n", /*PR_TOKEN_SEPARATORS,*/ Next);
670             if (!Token2)
671             {
672                 Token2 = "";
673             }
674 #endif
675             DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
676                 "New #define: %s->%s\n",
677                 Gbl_CurrentLineNumber, Token, Token2);

679             PrAddDefine (Token, Token2, FALSE);
680         }
681         break;

683     case PR_DIRECTIVE_ERROR:

685         /* Note: No macro expansion */

687         PrError (ASL_ERROR, ASL_MSG_ERROR_DIRECTIVE,
688             THIS_TOKEN_OFFSET (Token));

690         Gbl_SourceLine = 0;
691         Gbl_NextError = Gbl_ErrorLog;
692         CmCleanupAndExit ();
693         exit(1);

695     case PR_DIRECTIVE_INCLUDE:

697         Token = PrGetNextToken (NULL, " \"<>", Next);
698         if (!Token)
699         {
700             goto SyntaxError;
701         }

703         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
704             "Start #include file \"%s\"\n", Gbl_CurrentLineNumber,
705             Token, Gbl_CurrentLineNumber);

707         PrOpenIncludeFile (Token);
708         break;

710     case PR_DIRECTIVE_LINE:

712         TokenOffset = Token - Gbl_MainTokenBuffer;

714         Status = PrResolveIntegerExpression (
715             &Gbl_CurrentLineBuffer[TokenOffset-1], &Value);
716         if (ACPI_FAILURE (Status))
717         {
718             return;
719         }

721         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
```



```
722             "User #line invocation %s\n", Gbl_CurrentLineNumber,
723             Token);

725         /* Update local line numbers */

727         Gbl_CurrentLineNumber = (UINT32) Value;
728         Gbl_PreviousLineNumber = 0;

730         /* Emit #line into the preprocessor file */

732         FlPrintFile (ASL_FILE_PREPROCESSOR, "#line %u \"%s\"\n",
733             Gbl_CurrentLineNumber, Gbl_Files[ASL_FILE_INPUT].Filename);
734         break;

736     case PR_DIRECTIVE_PRAGMA:

738         if (!strcmp (Token, "disable"))
739         {
740             Token = PrGetNextToken (NULL, PR_TOKEN_SEPARATORS, Next);
741             if (!Token)
742             {
743                 goto SyntaxError;
744             }

746             TokenOffset = Token - Gbl_MainTokenBuffer;
747             AslDisableException (&Gbl_CurrentLineBuffer[TokenOffset]);
748         }
749         else if (!strcmp (Token, "message"))
750         {
751             Token = PrGetNextToken (NULL, PR_TOKEN_SEPARATORS, Next);
752             if (!Token)
753             {
754                 goto SyntaxError;
755             }

757             TokenOffset = Token - Gbl_MainTokenBuffer;
758             AcpiOsPrintf ("%s\n", &Gbl_CurrentLineBuffer[TokenOffset]);
759         }
760         else
761         {
762             PrError (ASL_ERROR, ASL_MSG_UNKNOWN_PRAGMA,
763                 THIS_TOKEN_OFFSET (Token));
764             return;
765         }

767         break;

769     case PR_DIRECTIVE_UNDEF:

771         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
772             "#undef: %s\n", Gbl_CurrentLineNumber, Token);

774         PrRemoveDefine (Token);
775         break;

777     case PR_DIRECTIVE_WARNING:

779         PrError (ASL_WARNING, ASL_MSG_WARNING_DIRECTIVE,
780             THIS_TOKEN_OFFSET (Token));
781         break;

783     default:

785         /* Should never get here */
786         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
787             "Unrecognized directive: %u\n",
```

```
788             Gbl_CurrentLineNumber, Directive);
789         break;
790     }

792     return;

794 SyntaxError:

796     PrError (ASL_ERROR, ASL_MSG_DIRECTIVE_SYNTAX,
797         THIS_TOKEN_OFFSET (DirectiveToken));
798     return;
799 }


802 /*******************************************************************************
803  *
804  * FUNCTION:    PrMatchDirective
805  *
806  * PARAMETERS:  Directive           - Pointer to directive name token
807  *
808  * RETURN:      Index into command array, -1 if not found
809  *
810  * DESCRIPTION: Lookup the incoming directive in the known directives table.
811  *
812  ******************************************************************************/

814 static int
815 PrMatchDirective (
816     char                    *Directive)
817 {
818     int                     i;


821     if (!Directive || Directive[0] == 0)
822     {
823         return (ASL_DIRECTIVE_NOT_FOUND);
824     }

826     for (i = 0; Gbl_DirectiveInfo[i].Name; i++)
827     {
828         if (!strcmp (Gbl_DirectiveInfo[i].Name, Directive))
829         {
830             return (i);
831         }
832     }

834     return (ASL_DIRECTIVE_NOT_FOUND);    /* Command not recognized */
835 }


838 /*******************************************************************************
839  *
840  * FUNCTION:    PrPushDirective
841  *
842  * PARAMETERS:  Directive           - Encoded directive ID
843  *              Argument            - String containing argument to the
844  *                                    directive
845  *
846  * RETURN:      None
847  *
848  * DESCRIPTION: Push an item onto the directive stack. Used for processing
849  *              nested #if/#else type conditional compilation directives.
850  *              Specifically: Used on detection of #if/#ifdef/#ifndef to open
851  *              a block.
852  *
853  ******************************************************************************/
```



```
855 static void
856 PrPushDirective (
857     int                     Directive,
858     char                    *Argument)
859 {
860     DIRECTIVE_INFO          *Info;


863     /* Allocate and populate a stack info item */

865     Info = ACPI_ALLOCATE (sizeof (DIRECTIVE_INFO));

867     Info->Next = Gbl_DirectiveStack;
868     Info->Directive = Directive;
869     Info->IgnoringThisCodeBlock = Gbl_IgnoringThisCodeBlock;
870     strncpy (Info->Argument, Argument, MAX_ARGUMENT_LENGTH);

872     DbgPrint (ASL_DEBUG_OUTPUT,
873         "Pr(%.4u) - [%u %s] %*s Pushed [#%s %s]: IgnoreFlag = %s\n",
874         Gbl_CurrentLineNumber, Gbl_IfDepth,
875         Gbl_IgnoringThisCodeBlock ? "I" : "E",
876         Gbl_IfDepth * 4, " ",
877         Gbl_DirectiveInfo[Directive].Name,
878         Argument, Gbl_IgnoringThisCodeBlock ? "TRUE" : "FALSE");

880     /* Push new item */

882     Gbl_DirectiveStack = Info;
883     Gbl_IfDepth++;
884 }


887 /*******************************************************************************
888  *
889  * FUNCTION:    PrPopDirective
890  *
891  * PARAMETERS:  None
892  *
893  * RETURN:      Status. Error if the stack is empty.
894  *
895  * DESCRIPTION: Pop an item off the directive stack. Used for processing
896  *              nested #if/#else type conditional compilation directives.
897  *              Specifically: Used on detection of #elif and #endif to remove
898  *              the original #if/#ifdef/#ifndef from the stack and close
899  *              the block.
900  *
901  ******************************************************************************/

903 static ACPI_STATUS
904 PrPopDirective (
905     void)
906 {
907     DIRECTIVE_INFO          *Info;


910     /* Check for empty stack */

912     Info = Gbl_DirectiveStack;
913     if (!Info)
914     {
915         return (AE_ERROR);
916     }

918     /* Pop one item, keep globals up-to-date */
```

```
    Gbl_IfDepth--;
    Gbl_IgnoringThisCodeBlock = Info->IgnoringThisCodeBlock;
    Gbl_DirectiveStack = Info->Next;

    DbgPrint (ASL_DEBUG_OUTPUT,
        "Pr(%.4u) - [%u %s] %*s Popped [#%s %s]: IgnoreFlag now = %s\n",
        Gbl_CurrentLineNumber, Gbl_IfDepth,
        Gbl_IgnoringThisCodeBlock ? "I" : "E",
        Gbl_IfDepth * 4, " ",
        Gbl_DirectiveInfo[Info->Directive].Name,
        Info->Argument, Gbl_IgnoringThisCodeBlock ? "TRUE" : "FALSE");

    ACPI_FREE (Info);
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    PrDbgPrint
 *
 * PARAMETERS:  Action              - Action being performed
 *              DirectiveName       - Directive being processed
 *
 * RETURN:      None
 *
 * DESCRIPTION: Special debug print for directive processing.
 *
 ******************************************************************************/

static void
PrDbgPrint (
    char                    *Action,
    char                    *DirectiveName)
{

    DbgPrint (ASL_DEBUG_OUTPUT, "Pr(%.4u) - [%u %s] "
        "%*s %s #%s, Depth %u\n",
        Gbl_CurrentLineNumber, Gbl_IfDepth,
        Gbl_IgnoringThisCodeBlock ? "I" : "E",
        Gbl_IfDepth * 4, " ",
        Action, DirectiveName, Gbl_IfDepth);
}
```

```
**********************************************************
   13064 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/compiler/prutils.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: prutils - Preprocessor utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aslcompiler.h"
#include "dtcompiler.h"


#define _COMPONENT          ASL_PREPROCESSOR
        ACPI_MODULE_NAME    ("prutils")


/******************************************************************************
 *
 * FUNCTION:    PrGetNextToken
 *
 * PARAMETERS:  Buffer              - Current line buffer
 *              MatchString         - String with valid token delimiters
 *              Next                - Set to next possible token in buffer
 *
 * RETURN:      Next token (null-terminated). Modifies the input line.
 *              Remainder of line is stored in *Next.
```



```
 *
 * DESCRIPTION: Local implementation of strtok() with local storage for the
 *              next pointer. Not only thread-safe, but allows multiple
 *              parsing of substrings such as expressions.
 *
 *****************************************************************************/

char *
PrGetNextToken (
    char                    *Buffer,
    char                    *MatchString,
    char                    **Next)
{
    char                    *TokenStart;


    if (!Buffer)
    {
        /* Use Next if it is valid */

        Buffer = *Next;
        if (!(*Next))
        {
            return (NULL);
        }
    }

    /* Skip any leading delimiters */

    while (*Buffer)
    {
        if (strchr (MatchString, *Buffer))
        {
            Buffer++;
        }
        else
        {
            break;
        }
    }

    /* Anything left on the line? */

    if (!(*Buffer))
    {
        *Next = NULL;
        return (NULL);
    }

    TokenStart = Buffer;

    /* Find the end of this token */

    while (*Buffer)
    {
        if (strchr (MatchString, *Buffer))
        {
            *Buffer = 0;
            *Next = Buffer+1;
            if (!**Next)
            {
                *Next = NULL;
            }
            return (TokenStart);
        }
        Buffer++;
```

```
128     }

130     *Next = NULL;
131     return (TokenStart);
132 }


135 /******************************************************************************
136  *
137  * FUNCTION:    PrError
138  *
139  * PARAMETERS:  Level               - Seriousness (Warning/error, etc.)
140  *              MessageId           - Index into global message buffer
141  *              Column              - Column in current line
142  *
143  * RETURN:      None
144  *
145  * DESCRIPTION: Preprocessor error reporting. Front end to AslCommonError2
146  *
147  *****************************************************************************/

149 void
150 PrError (
151     UINT8                   Level,
152     UINT8                   MessageId,
153     UINT32                  Column)
154 {
155 #if 0
156     AcpiOsPrintf ("%s (%u) : %s", Gbl_Files[ASL_FILE_INPUT].Filename,
157         Gbl_CurrentLineNumber, Gbl_CurrentLineBuffer);
158 #endif


161     if (Column > 120)
162     {
163         Column = 0;
164     }

166     /* TBD: Need Logical line number? */

168     AslCommonError2 (Level, MessageId,
169         Gbl_CurrentLineNumber, Column,
170         Gbl_CurrentLineBuffer,
171         Gbl_Files[ASL_FILE_INPUT].Filename, "Preprocessor");

173     Gbl_PreprocessorError = TRUE;
174 }


177 /******************************************************************************
178  *
179  * FUNCTION:    PrReplaceData
180  *
181  * PARAMETERS:  Buffer              - Original(target) buffer pointer
182  *              LengthToRemove      - Length to be removed from target buffer
183  *              BufferToAdd         - Data to be inserted into target buffer
184  *              LengthToAdd         - Length of BufferToAdd
185  *
186  * RETURN:      None
187  *
188  * DESCRIPTION: Generic buffer data replacement.
189  *
190  *****************************************************************************/

192 void
193 PrReplaceData (
```



```
194     char                    *Buffer,
195     UINT32                  LengthToRemove,
196     char                    *BufferToAdd,
197     UINT32                  LengthToAdd)
198 {
199     UINT32                  BufferLength;


202     /* Buffer is a string, so the length must include the terminating zero */

204     BufferLength = strlen (Buffer) + 1;

206     if (LengthToRemove != LengthToAdd)
207     {
208         /*
209          * Move some of the existing data
210          * 1) If adding more bytes than removing, make room for the new data
211          * 2) if removing more bytes than adding, delete the extra space
212          */
213         if (LengthToRemove > 0)
214         {
215             memmove ((Buffer + LengthToAdd), (Buffer + LengthToRemove),
216                 (BufferLength - LengthToRemove));
217         }
218     }

220     /* Now we can move in the new data */

222     if (LengthToAdd > 0)
223     {
224         memmove (Buffer, BufferToAdd, LengthToAdd);
225     }
226 }


229 /******************************************************************************
230  *
231  * FUNCTION:    PrOpenIncludeFile
232  *
233  * PARAMETERS:  Filename            - Filename or pathname for include file
234  *
235  * RETURN:      None.
236  *
237  * DESCRIPTION: Open an include file and push it on the input file stack.
238  *
239  *****************************************************************************/

241 void
242 PrOpenIncludeFile (
243     char                    *Filename)
244 {
245     FILE                    *IncludeFile;
246     ASL_INCLUDE_DIR         *NextDir;


249     /* Start the actual include file on the next line */

251     Gbl_CurrentLineOffset++;

253     /* Attempt to open the include file */
254     /* If the file specifies an absolute path, just open it */

256     if ((Filename[0] == '/') ||
257         (Filename[0] == '\\') ||
258         (Filename[1] == ':'))
259     {
```

```
260         IncludeFile = PrOpenIncludeWithPrefix ("", Filename);
261         if (!IncludeFile)
262         {
263             goto ErrorExit;
264         }
265         return;
266     }
267
268     /*
269      * The include filename is not an absolute path.
270      *
271      * First, search for the file within the "local" directory -- meaning
272      * the same directory that contains the source file.
273      *
274      * Construct the file pathname from the global directory name.
275      */
276     IncludeFile = PrOpenIncludeWithPrefix (Gbl_DirectoryPath, Filename);
277     if (IncludeFile)
278     {
279         return;
280     }
281
282     /*
283      * Second, search for the file within the (possibly multiple)
284      * directories specified by the -I option on the command line.
285      */
286     NextDir = Gbl_IncludeDirList;
287     while (NextDir)
288     {
289         IncludeFile = PrOpenIncludeWithPrefix (NextDir->Dir, Filename);
290         if (IncludeFile)
291         {
292             return;
293         }
294
295         NextDir = NextDir->Next;
296     }
297
298     /* We could not open the include file after trying very hard */
299
300 ErrorExit:
301     sprintf (Gbl_MainTokenBuffer, "%s, %s", Filename, strerror (errno));
302     PrError (ASL_ERROR, ASL_MSG_INCLUDE_FILE_OPEN, 0);
303 }
304
305
306 /******************************************************************************
307  *
308  * FUNCTION:    FlOpenIncludeWithPrefix
309  *
310  * PARAMETERS:  PrefixDir       - Prefix directory pathname. Can be a zero
311  *                                length string.
312  *              Filename        - The include filename from the source ASL.
313  *
314  * RETURN:      Valid file descriptor if successful. Null otherwise.
315  *
316  * DESCRIPTION: Open an include file and push it on the input file stack.
317  *
318  *****************************************************************************/
319
320 FILE *
321 PrOpenIncludeWithPrefix (
322     char                    *PrefixDir,
323     char                    *Filename)
324 {
325     FILE                    *IncludeFile;
```



```
326     char                    *Pathname;
327
328
329     /* Build the full pathname to the file */
330
331     Pathname = FlMergePathnames (PrefixDir, Filename);
332
333     DbgPrint (ASL_PARSE_OUTPUT, PR_PREFIX_ID
334         "Include: Opening file - \"%s\"\n",
335         Gbl_CurrentLineNumber, Pathname);
336
337     /* Attempt to open the file, push if successful */
338
339     IncludeFile = fopen (Pathname, "r");
340     if (!IncludeFile)
341     {
342         fprintf (stderr, "Could not open include file %s\n", Pathname);
343         ACPI_FREE (Pathname);
344         return (NULL);
345     }
346
347     /* Push the include file on the open input file stack */
348
349     PrPushInputFileStack (IncludeFile, Pathname);
350     return (IncludeFile);
351 }
352
353
354 /******************************************************************************
355  *
356  * FUNCTION:    AslPushInputFileStack
357  *
358  * PARAMETERS:  InputFile           - Open file pointer
359  *              Filename            - Name of the file
360  *
361  * RETURN:      None
362  *
363  * DESCRIPTION: Push the InputFile onto the file stack, and point the parser
364  *              to this file. Called when an include file is successfully
365  *              opened.
366  *
367  *****************************************************************************/
368
369 void
370 PrPushInputFileStack (
371     FILE                    *InputFile,
372     char                    *Filename)
373 {
374     PR_FILE_NODE            *Fnode;
375
376
377     /* Save the current state in an Fnode */
378
379     Fnode = UtLocalCalloc (sizeof (PR_FILE_NODE));
380
381     Fnode->File = Gbl_Files[ASL_FILE_INPUT].Handle;
382     Fnode->Next = Gbl_InputFileList;
383     Fnode->Filename = Gbl_Files[ASL_FILE_INPUT].Filename;
384     Fnode->CurrentLineNumber = Gbl_CurrentLineNumber;
385
386     /* Push it on the stack */
387
388     Gbl_InputFileList = Fnode;
389
390     DbgPrint (ASL_PARSE_OUTPUT, PR_PREFIX_ID
391         "Push InputFile Stack: handle %p\n\n",
```

```
392         Gbl_CurrentLineNumber, InputFile);

394     /* Reset the global line count and filename */

396     Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
397     Gbl_Files[ASL_FILE_INPUT].Handle = InputFile;
398     Gbl_PreviousLineNumber = 0;
399     Gbl_CurrentLineNumber = 0;

401     /* Emit a new #line directive for the include file */

403     FlPrintFile (ASL_FILE_PREPROCESSOR, "#line %u \"%s\"\n",
404         1, Filename);
405 }


408 /******************************************************************************
409  *
410  * FUNCTION:    AslPopInputFileStack
411  *
412  * PARAMETERS:  None
413  *
414  * RETURN:      0 if a node was popped, -1 otherwise
415  *
416  * DESCRIPTION: Pop the top of the input file stack and point the parser to
417  *              the saved parse buffer contained in the fnode. Also, set the
418  *              global line counters to the saved values. This function is
419  *              called when an include file reaches EOF.
420  *
421  *****************************************************************************/

423 BOOLEAN
424 PrPopInputFileStack (
425     void)
426 {
427     PR_FILE_NODE            *Fnode;


430     Fnode = Gbl_InputFileList;
431     DbgPrint (ASL_PARSE_OUTPUT, "\n" PR_PREFIX_ID
432         "Pop InputFile Stack, Fnode %p\n\n",
433         Gbl_CurrentLineNumber, Fnode);

435     if (!Fnode)
436     {
437         return (FALSE);
438     }

440     /* Close the current include file */

442     fclose (Gbl_Files[ASL_FILE_INPUT].Handle);

444     /* Update the top-of-stack */

446     Gbl_InputFileList = Fnode->Next;

448     /* Reset global line counter and filename */

450     Gbl_Files[ASL_FILE_INPUT].Filename = Fnode->Filename;
451     Gbl_Files[ASL_FILE_INPUT].Handle = Fnode->File;
452     Gbl_CurrentLineNumber = Fnode->CurrentLineNumber;
453     Gbl_PreviousLineNumber = 0;

455     /* Emit a new #line directive after the include file */

457     FlPrintFile (ASL_FILE_PREPROCESSOR, "#line %u \"%s\"\n",
```



```
458         Gbl_CurrentLineNumber + 1, Fnode->Filename);

460     /* All done with this node */

462     ACPI_FREE (Fnode);
463     return (TRUE);
464 }
```

```
**********************************************************
    4383 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/compiler/readme.txt
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/*
 * Miscellaneous instructions for building and using the iASL compiler.
 */
Last update 9 December 2013.


1) Generating iASL from source
------------------------------

Generation of the ASL compiler from source code requires these items:

    1) The ACPICA source code tree.
    2) An ANSI C compiler.
    3) The Flex (or Lex) lexical analyzer generator.
    4) The Bison (or Yacc) parser generator.

There are three major ACPICA source code components that are required to
generate the compiler (Basically, the entire ACPICA source tree should
be installed):

    1) The ASL compiler source.
    2) The ACPICA Core Subsystem source. In particular, the Namespace
       Manager component is used to create an internal ACPI namespace
       and symbol table, and the AML Interpreter is used to evaluate
       constant expressions.
    3) The "common" source directory that is used for all ACPI components.


1a) Notes for Linux/Unix generation
-----------------------------------

iASL has been generated with these versions of Flex/Bison:

    flex:  Version 2.5.32
    bison: Version 2.6.2

Other required packages:

    make
    gcc C compiler
    m4 (macro processor required by bison)

On Linux/Unix systems, the following commands will build the compiler:

    cd acpica (or cd acpica/generate/unix)
    make clean
    make iasl


1b) Notes for Windows generation
--------------------------------

On Windows, the Visual Studio 2008 project file appears in this directory:

    generate/msvc9/AcpiComponents.sln

The Windows versions of GNU Flex/Bison must be installed, and they must
be installed in a directory that contains no embedded spaces in the
pathname. They cannot be installed in the default "c:\Program Files"
directory. This is a bug in Bison. The default Windows project file for
```



```
iASL assumes that these tools are installed at this location:

    c:\GnuWin32

Once the tools are installed, ensure that this path is added to the
default system $Path environment variable:

    c:\GnuWin32\bin

Goto: ControlPanel/System/AdvancedSystemSettings/EnvironmentVariables

Important: Now Windows must be rebooted to make the system aware of
the updated $Path. Otherwise, Bison will not be able to find the M4
interpreter library and will fail.

iASL has been generated with these versions of Flex/Bison for Windows:

    Flex for Windows:  V2.5.4a
    Bison for Windows: V2.4.1

Flex is available at:  http://gnuwin32.sourceforge.net/packages/flex.htm
Bison is available at: http://gnuwin32.sourceforge.net/packages/bison.htm



2) Integration as a custom tool for Visual Studio
-------------------------------------------------

This procedure adds the iASL compiler as a custom tool that can be used
to compile ASL source files. The output is sent to the VC output
window.

a) Select Tools->Customize.

b) Select the "Tools" tab.

c) Scroll down to the bottom of the "Menu Contents" window. There you
   will see an empty rectangle. Click in the rectangle to enter a
   name for this tool.

d) Type "iASL Compiler" in the box and hit enter. You can now edit
   the other fields for this new custom tool.

e) Enter the following into the fields:

   Command:             C:\Acpi\iasl.exe
   Arguments:           -vi "$(FilePath)"
   Initial Directory    "$(FileDir)"
   Use Output Window    <Check this option>

   "Command" must be the path to wherever you copied the compiler.
   "-vi" instructs the compiler to produce messages appropriate for VC.
   Quotes around FilePath and FileDir enable spaces in filenames.

f) Select "Close".

These steps will add the compiler to the tools menu as a custom tool.
By enabling "Use Output Window", you can click on error messages in
the output window and the source file and source line will be
automatically displayed by VC. Also, you can use F4 to step through
the messages and the corresponding source line(s).



3) Integrating iASL into a Visual Studio ASL project build
----------------------------------------------------------
```

```
128 This procedure creates a project that compiles ASL files to AML.

130 a) Create a new, empty project and add your .ASL files to the project

132 b) For all ASL files in the project, specify a custom build (under
133 Project/Settings/CustomBuild with the following settings (or similar):

135 Commands:
136     c:\acpi\libraries\iasl.exe -vs -vi "$(InputPath)"

138 Output:
139     $(InputDir)\$(InputPath).aml
```

```
**********************************************************
   33864 Thu Dec 26 13:48:40 2013
new/usr/src/common/acpica/components/debugger/dbcmds.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbcmds - Miscellaneous debug commands and output routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acdebug.h"
#include "acnamesp.h"
#include "acresrc.h"
#include "actables.h"

#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbcmds")


/* Local prototypes */
```



```
static void
AcpiDmCompareAmlResources (
    UINT8                   *Aml1Buffer,
    ACPI_RSDESC_SIZE        Aml1BufferLength,
    UINT8                   *Aml2Buffer,
    ACPI_RSDESC_SIZE        Aml2BufferLength);

static ACPI_STATUS
AcpiDmTestResourceConversion (
    ACPI_NAMESPACE_NODE     *Node,
    char                    *Name);

static ACPI_STATUS
AcpiDbResourceCallback (
    ACPI_RESOURCE           *Resource,
    void                    *Context);

static ACPI_STATUS
AcpiDbDeviceResources (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);

static void
AcpiDbDoOneSleepState (
    UINT8                   SleepState);


/******************************************************************************
 *
 * FUNCTION:    AcpiDbConvertToNode
 *
 * PARAMETERS:  InString            - String to convert
 *
 * RETURN:      Pointer to a NS node
 *
 * DESCRIPTION: Convert a string to a valid NS pointer. Handles numeric or
 *              alphanumeric strings.
 *
 *****************************************************************************/

ACPI_NAMESPACE_NODE *
AcpiDbConvertToNode (
    char                    *InString)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_SIZE               Address;


    if ((*InString >= 0x30) && (*InString <= 0x39))
    {
        /* Numeric argument, convert */

        Address = ACPI_STRTOUL (InString, NULL, 16);
        Node = ACPI_TO_POINTER (Address);
        if (!AcpiOsReadable (Node, sizeof (ACPI_NAMESPACE_NODE)))
        {
            AcpiOsPrintf ("Address %p is invalid in this address space\n",
                Node);
            return (NULL);
        }

        /* Make sure pointer is valid NS node */
```

```
126         if (ACPI_GET_DESCRIPTOR_TYPE (Node) != ACPI_DESC_TYPE_NAMED)
127         {
128             AcpiOsPrintf ("Address %p is not a valid NS node [%s]\n",
129                     Node, AcpiUtGetDescriptorName (Node));
130             return (NULL);
131         }
132     }
133     else
134     {
135         /*
136          * Alpha argument: The parameter is a name string that must be
137          * resolved to a Namespace object.
138          */
139         Node = AcpiDbLocalNsLookup (InString);
140         if (!Node)
141         {
142             Node = AcpiGbl_RootNode;
143         }
144     }

146     return (Node);
147 }


150 /******************************************************************************
151  *
152  * FUNCTION:    AcpiDbSleep
153  *
154  * PARAMETERS:  ObjectArg           - Desired sleep state (0-5). NULL means
155  *                                    invoke all possible sleep states.
156  *
157  * RETURN:      Status
158  *
159  * DESCRIPTION: Simulate sleep/wake sequences
160  *
161  *****************************************************************************/

163 ACPI_STATUS
164 AcpiDbSleep (
165     char                    *ObjectArg)
166 {
167     UINT8                   SleepState;
168     UINT32                  i;


171     ACPI_FUNCTION_TRACE (AcpiDbSleep);


174     /* Null input (no arguments) means to invoke all sleep states */

176     if (!ObjectArg)
177     {
178         AcpiOsPrintf ("Invoking all possible sleep states, 0-%d\n",
179             ACPI_S_STATES_MAX);

181         for (i = 0; i <= ACPI_S_STATES_MAX; i++)
182         {
183             AcpiDbDoOneSleepState ((UINT8) i);
184         }

186         return_ACPI_STATUS (AE_OK);
187     }

189     /* Convert argument to binary and invoke the sleep state */

191     SleepState = (UINT8) ACPI_STRTOUL (ObjectArg, NULL, 0);
```



```
192     AcpiDbDoOneSleepState (SleepState);
193     return_ACPI_STATUS (AE_OK);
194 }


197 /******************************************************************************
198  *
199  * FUNCTION:    AcpiDbDoOneSleepState
200  *
201  * PARAMETERS:  SleepState          - Desired sleep state (0-5)
202  *
203  * RETURN:      Status
204  *
205  * DESCRIPTION: Simulate a sleep/wake sequence
206  *
207  *****************************************************************************/

209 static void
210 AcpiDbDoOneSleepState (
211     UINT8                   SleepState)
212 {
213     ACPI_STATUS             Status;
214     UINT8                   SleepTypeA;
215     UINT8                   SleepTypeB;


218     /* Validate parameter */

220     if (SleepState > ACPI_S_STATES_MAX)
221     {
222         AcpiOsPrintf ("Sleep state %d out of range (%d max)\n",
223             SleepState, ACPI_S_STATES_MAX);
224         return;
225     }

227     AcpiOsPrintf ("\n---- Invoking sleep state S%d (%s):\n",
228         SleepState, AcpiGbl_SleepStateNames[SleepState]);

230     /* Get the values for the sleep type registers (for display only) */

232     Status = AcpiGetSleepTypeData (SleepState, &SleepTypeA, &SleepTypeB);
233     if (ACPI_FAILURE (Status))
234     {
235         AcpiOsPrintf ("Could not evaluate [%s] method, %s\n",
236             AcpiGbl_SleepStateNames[SleepState],
237             AcpiFormatException (Status));
238         return;
239     }

241     AcpiOsPrintf (
242         "Register values for sleep state S%d: Sleep-A: %.2X, Sleep-B: %.2X\n",
243         SleepState, SleepTypeA, SleepTypeB);

245     /* Invoke the various sleep/wake interfaces */

247     AcpiOsPrintf ("**** Sleep: Prepare to sleep (S%d) ****\n",
248         SleepState);
249     Status = AcpiEnterSleepStatePrep (SleepState);
250     if (ACPI_FAILURE (Status))
251     {
252         goto ErrorExit;
253     }

255     AcpiOsPrintf ("**** Sleep: Going to sleep (S%d) ****\n",
256         SleepState);
257     Status = AcpiEnterSleepState (SleepState);
```

```
    if (ACPI_FAILURE (Status))
    {
        goto ErrorExit;
    }

    AcpiOsPrintf ("**** Wake: Prepare to return from sleep (S%d) ****\n",
        SleepState);
    Status = AcpiLeaveSleepStatePrep (SleepState);
    if (ACPI_FAILURE (Status))
    {
        goto ErrorExit;
    }

    AcpiOsPrintf ("**** Wake: Return from sleep (S%d) ****\n",
        SleepState);
    Status = AcpiLeaveSleepState (SleepState);
    if (ACPI_FAILURE (Status))
    {
        goto ErrorExit;
    }

    return;


ErrorExit:
    ACPI_EXCEPTION ((AE_INFO, Status, "During invocation of sleep state S%d",
        SleepState));
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDbDisplayLocks
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display information about internal mutexes.
 *
 ******************************************************************************/

void
AcpiDbDisplayLocks (
    void)
{
    UINT32                  i;


    for (i = 0; i < ACPI_MAX_MUTEX; i++)
    {
        AcpiOsPrintf ("%26s : %s\n", AcpiUtGetMutexName (i),
            AcpiGbl_MutexInfo[i].ThreadId == ACPI_MUTEX_NOT_ACQUIRED
                ? "Locked" : "Unlocked");
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDbDisplayTableInfo
 *
 * PARAMETERS:  TableArg            - Name of table to be displayed
 *
 * RETURN:      None
 *
```



```
 * DESCRIPTION: Display information about loaded tables. Current
 *              implementation displays all loaded tables.
 *
 ******************************************************************************/

void
AcpiDbDisplayTableInfo (
    char                    *TableArg)
{
    UINT32                  i;
    ACPI_TABLE_DESC         *TableDesc;
    ACPI_STATUS             Status;


    /* Header */

    AcpiOsPrintf ("Idx ID Status    Type            Sig  Address  Len   Header\n

    /* Walk the entire root table list */

    for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
    {
        TableDesc = &AcpiGbl_RootTableList.Tables[i];

        /* Index and Table ID */

        AcpiOsPrintf ("%3u %.2u ", i, TableDesc->OwnerId);

        /* Decode the table flags */

        if (!(TableDesc->Flags & ACPI_TABLE_IS_LOADED))
        {
            AcpiOsPrintf ("NotLoaded ");
        }
        else
        {
            AcpiOsPrintf ("   Loaded ");
        }

        switch (TableDesc->Flags & ACPI_TABLE_ORIGIN_MASK)
        {
        case ACPI_TABLE_ORIGIN_UNKNOWN:

            AcpiOsPrintf ("Unknown   ");
            break;

        case ACPI_TABLE_ORIGIN_MAPPED:

            AcpiOsPrintf ("Mapped    ");
            break;

        case ACPI_TABLE_ORIGIN_ALLOCATED:

            AcpiOsPrintf ("Allocated ");
            break;

        case ACPI_TABLE_ORIGIN_OVERRIDE:

            AcpiOsPrintf ("Override  ");
            break;

        default:

            AcpiOsPrintf ("INVALID   ");
            break;
        }
```

```
391         /* Make sure that the table is mapped */

393         Status = AcpiTbVerifyTable (TableDesc);
394         if (ACPI_FAILURE (Status))
395         {
396             return;
397         }

399         /* Dump the table header */

401         if (TableDesc->Pointer)
402         {
403             AcpiTbPrintTableHeader (TableDesc->Address, TableDesc->Pointer);
404         }
405         else
406         {
407             /* If the pointer is null, the table has been unloaded */

409             ACPI_INFO ((AE_INFO, "%4.4s - Table has been unloaded",
410                 TableDesc->Signature.Ascii));
411         }
412     }
413 }


416 /******************************************************************************
417  *
418  * FUNCTION:    AcpiDbUnloadAcpiTable
419  *
420  * PARAMETERS:  ObjectName          - Namespace pathname for an object that
421  *                                    is owned by the table to be unloaded
422  *
423  * RETURN:      None
424  *
425  * DESCRIPTION: Unload an ACPI table, via any namespace node that is owned
426  *              by the table.
427  *
428  *****************************************************************************/

430 void
431 AcpiDbUnloadAcpiTable (
432     char                    *ObjectName)
433 {
434     ACPI_NAMESPACE_NODE     *Node;
435     ACPI_STATUS             Status;


438     /* Translate name to an Named object */

440     Node = AcpiDbConvertToNode (ObjectName);
441     if (!Node)
442     {
443         AcpiOsPrintf ("Could not find [%s] in namespace\n",
444             ObjectName);
445         return;
446     }

448     Status = AcpiUnloadParentTable (ACPI_CAST_PTR (ACPI_HANDLE, Node));
449     if (ACPI_SUCCESS (Status))
450     {
451         AcpiOsPrintf ("Parent of [%s] (%p) unloaded and uninstalled\n",
452             ObjectName, Node);
453     }
454     else
455     {
```



```
456         AcpiOsPrintf ("%s, while unloading parent table of [%s]\n",
457             AcpiFormatException (Status), ObjectName);
458     }
459 }


462 /******************************************************************************
463  *
464  * FUNCTION:    AcpiDbSendNotify
465  *
466  * PARAMETERS:  Name                - Name of ACPI object where to send notify
467  *              Value               - Value of the notify to send.
468  *
469  * RETURN:      None
470  *
471  * DESCRIPTION: Send an ACPI notification. The value specified is sent to the
472  *              named object as an ACPI notify.
473  *
474  *****************************************************************************/

476 void
477 AcpiDbSendNotify (
478     char                    *Name,
479     UINT32                  Value)
480 {
481     ACPI_NAMESPACE_NODE     *Node;
482     ACPI_STATUS             Status;


485     /* Translate name to an Named object */

487     Node = AcpiDbConvertToNode (Name);
488     if (!Node)
489     {
490         return;
491     }

493     /* Dispatch the notify if legal */

495     if (AcpiEvIsNotifyObject (Node))
496     {
497         Status = AcpiEvQueueNotifyRequest (Node, Value);
498         if (ACPI_FAILURE (Status))
499         {
500             AcpiOsPrintf ("Could not queue notify\n");
501         }
502     }
503     else
504     {
505         AcpiOsPrintf (
506             "Named object [%4.4s] Type %s, must be Device/Thermal/Processor type
507             AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type));
508     }
509 }


512 /******************************************************************************
513  *
514  * FUNCTION:    AcpiDbDisplayInterfaces
515  *
516  * PARAMETERS:  ActionArg           - Null, "install", or "remove"
517  *              InterfaceNameArg    - Name for install/remove options
518  *
519  * RETURN:      None
520  *
521  * DESCRIPTION: Display or modify the global _OSI interface list
```

```
 *
 *****************************************************************************/

void
AcpiDbDisplayInterfaces (
    char                    *ActionArg,
    char                    *InterfaceNameArg)
{
    ACPI_INTERFACE_INFO     *NextInterface;
    char                    *SubString;
    ACPI_STATUS             Status;


    /* If no arguments, just display current interface list */

    if (!ActionArg)
    {
        (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex,
                    ACPI_WAIT_FOREVER);

        NextInterface = AcpiGbl_SupportedInterfaces;
        while (NextInterface)
        {
            if (!(NextInterface->Flags & ACPI_OSI_INVALID))
            {
                AcpiOsPrintf ("%s\n", NextInterface->Name);
            }
            NextInterface = NextInterface->Next;
        }

        AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
        return;
    }

    /* If ActionArg exists, so must InterfaceNameArg */

    if (!InterfaceNameArg)
    {
        AcpiOsPrintf ("Missing Interface Name argument\n");
        return;
    }

    /* Uppercase the action for match below */

    AcpiUtStrupr (ActionArg);

    /* Install - install an interface */

    SubString = ACPI_STRSTR ("INSTALL", ActionArg);
    if (SubString)
    {
        Status = AcpiInstallInterface (InterfaceNameArg);
        if (ACPI_FAILURE (Status))
        {
            AcpiOsPrintf ("%s, while installing \"%s\"\n",
                AcpiFormatException (Status), InterfaceNameArg);
        }
        return;
    }

    /* Remove - remove an interface */

    SubString = ACPI_STRSTR ("REMOVE", ActionArg);
    if (SubString)
    {
        Status = AcpiRemoveInterface (InterfaceNameArg);
```



```
        if (ACPI_FAILURE (Status))
        {
            AcpiOsPrintf ("%s, while removing \"%s\"\n",
                AcpiFormatException (Status), InterfaceNameArg);
        }
        return;
    }

    /* Invalid ActionArg */

    AcpiOsPrintf ("Invalid action argument: %s\n", ActionArg);
    return;
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbDisplayTemplate
 *
 * PARAMETERS:  BufferArg           - Buffer name or address
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump a buffer that contains a resource template
 *
 ****************************************************************************/

void
AcpiDbDisplayTemplate (
    char                    *BufferArg)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;
    ACPI_BUFFER             ReturnBuffer;


    /* Translate BufferArg to an Named object */

    Node = AcpiDbConvertToNode (BufferArg);
    if (!Node || (Node == AcpiGbl_RootNode))
    {
        AcpiOsPrintf ("Invalid argument: %s\n", BufferArg);
        return;
    }

    /* We must have a buffer object */

    if (Node->Type != ACPI_TYPE_BUFFER)
    {
        AcpiOsPrintf ("Not a Buffer object, cannot be a template: %s\n",
            BufferArg);
        return;
    }

    ReturnBuffer.Length = ACPI_DEBUG_BUFFER_SIZE;
    ReturnBuffer.Pointer = AcpiGbl_DbBuffer;

    /* Attempt to convert the raw buffer to a resource list */

    Status = AcpiRsCreateResourceList (Node->Object, &ReturnBuffer);

    AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
    AcpiDbgLevel |= ACPI_LV_RESOURCES;

    if (ACPI_FAILURE (Status))
    {
```

```
654         AcpiOsPrintf ("Could not convert Buffer to a resource list: %s, %s\n",
655             BufferArg, AcpiFormatException (Status));
656         goto DumpBuffer;
657     }

659     /* Now we can dump the resource list */

661     AcpiRsDumpResourceList (ACPI_CAST_PTR (ACPI_RESOURCE,
662         ReturnBuffer.Pointer));

664 DumpBuffer:
665     AcpiOsPrintf ("\nRaw data buffer:\n");
666     AcpiUtDebugDumpBuffer ((UINT8 *) Node->Object->Buffer.Pointer,
667         Node->Object->Buffer.Length,
668         DB_BYTE_DISPLAY, ACPI_UINT32_MAX);

670     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
671     return;
672 }


675 /*******************************************************************************
676  *
677  * FUNCTION:    AcpiDmCompareAmlResources
678  *
679  * PARAMETERS:  Aml1Buffer          - Contains first resource list
680  *              Aml1BufferLength    - Length of first resource list
681  *              Aml2Buffer          - Contains second resource list
682  *              Aml2BufferLength    - Length of second resource list
683  *
684  * RETURN:      None
685  *
686  * DESCRIPTION: Compare two AML resource lists, descriptor by descriptor (in
687  *              order to isolate a miscompare to an individual resource)
688  *
689  ******************************************************************************/

691 static void
692 AcpiDmCompareAmlResources (
693     UINT8                   *Aml1Buffer,
694     ACPI_RSDESC_SIZE        Aml1BufferLength,
695     UINT8                   *Aml2Buffer,
696     ACPI_RSDESC_SIZE        Aml2BufferLength)
697 {
698     UINT8                   *Aml1;
699     UINT8                   *Aml2;
700     UINT8                   *Aml1End;
701     UINT8                   *Aml2End;
702     ACPI_RSDESC_SIZE        Aml1Length;
703     ACPI_RSDESC_SIZE        Aml2Length;
704     ACPI_RSDESC_SIZE        Offset = 0;
705     UINT8                   ResourceType;
706     UINT32                  Count = 0;
707     UINT32                  i;


710     /* Compare overall buffer sizes (may be different due to size rounding) */

712     if (Aml1BufferLength != Aml2BufferLength)
713     {
714         AcpiOsPrintf (
715             "**** Buffer length mismatch in converted AML: Original %X, New %X *
716             Aml1BufferLength, Aml2BufferLength);
717     }

719     Aml1 = Aml1Buffer;
```



```
720     Aml2 = Aml2Buffer;
721     Aml1End = Aml1Buffer + Aml1BufferLength;
722     Aml2End = Aml2Buffer + Aml2BufferLength;

724     /* Walk the descriptor lists, comparing each descriptor */

726     while ((Aml1 < Aml1End) && (Aml2 < Aml2End))
727     {
728         /* Get the lengths of each descriptor */

730         Aml1Length = AcpiUtGetDescriptorLength (Aml1);
731         Aml2Length = AcpiUtGetDescriptorLength (Aml2);
732         ResourceType = AcpiUtGetResourceType (Aml1);

734         /* Check for descriptor length match */

736         if (Aml1Length != Aml2Length)
737         {
738             AcpiOsPrintf (
739                 "**** Length mismatch in descriptor [%.2X] type %2.2X, Offset %8
740                 Count, ResourceType, Offset, Aml1Length, Aml2Length);
741         }

743         /* Check for descriptor byte match */

745         else if (ACPI_MEMCMP (Aml1, Aml2, Aml1Length))
746         {
747             AcpiOsPrintf (
748                 "**** Data mismatch in descriptor [%.2X] type %2.2X, Offset %8.8
749                 Count, ResourceType, Offset);

751             for (i = 0; i < Aml1Length; i++)
752             {
753                 if (Aml1[i] != Aml2[i])
754                 {
755                     AcpiOsPrintf (
756                         "Mismatch at byte offset %.2X: is %2.2X, should be %2.2X
757                         i, Aml2[i], Aml1[i]);
758                 }
759             }
760         }

762         /* Exit on EndTag descriptor */

764         if (ResourceType == ACPI_RESOURCE_NAME_END_TAG)
765         {
766             return;
767         }

769         /* Point to next descriptor in each buffer */

771         Count++;
772         Offset += Aml1Length;
773         Aml1 += Aml1Length;
774         Aml2 += Aml2Length;
775     }
776 }


779 /*******************************************************************************
780  *
781  * FUNCTION:    AcpiDmTestResourceConversion
782  *
783  * PARAMETERS:  Node                - Parent device node
784  *              Name                - resource method name (_CRS)
785  *
```

```
 * RETURN:      Status
 *
 * DESCRIPTION: Compare the original AML with a conversion of the AML to
 *              internal resource list, then back to AML.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDmTestResourceConversion (
    ACPI_NAMESPACE_NODE     *Node,
    char                    *Name)
{
    ACPI_STATUS             Status;
    ACPI_BUFFER             ReturnBuffer;
    ACPI_BUFFER             ResourceBuffer;
    ACPI_BUFFER             NewAml;
    ACPI_OBJECT             *OriginalAml;


    AcpiOsPrintf ("Resource Conversion Comparison:\n");

    NewAml.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
    ReturnBuffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
    ResourceBuffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;

    /* Get the original _CRS AML resource template */

    Status = AcpiEvaluateObject (Node, Name, NULL, &ReturnBuffer);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("Could not obtain %s: %s\n",
            Name, AcpiFormatException (Status));
        return (Status);
    }

    /* Get the AML resource template, converted to internal resource structs */

    Status = AcpiGetCurrentResources (Node, &ResourceBuffer);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("AcpiGetCurrentResources failed: %s\n",
            AcpiFormatException (Status));
        goto Exit1;
    }

    /* Convert internal resource list to external AML resource template */

    Status = AcpiRsCreateAmlResources (&ResourceBuffer, &NewAml);
    if (ACPI_FAILURE (Status))
    {
        AcpiOsPrintf ("AcpiRsCreateAmlResources failed: %s\n",
            AcpiFormatException (Status));
        goto Exit2;
    }

    /* Compare original AML to the newly created AML resource list */

    OriginalAml = ReturnBuffer.Pointer;

    AcpiDmCompareAmlResources (OriginalAml->Buffer.Pointer,
        (ACPI_RSDESC_SIZE) OriginalAml->Buffer.Length,
        NewAml.Pointer, (ACPI_RSDESC_SIZE) NewAml.Length);

    /* Cleanup and exit */

    ACPI_FREE (NewAml.Pointer);
```



```
Exit2:
    ACPI_FREE (ResourceBuffer.Pointer);
Exit1:
    ACPI_FREE (ReturnBuffer.Pointer);
    return (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbResourceCallback
 *
 * PARAMETERS:  ACPI_WALK_RESOURCE_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Simple callback to exercise AcpiWalkResources and
 *              AcpiWalkResourceBuffer.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDbResourceCallback (
    ACPI_RESOURCE           *Resource,
    void                    *Context)
{

    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbDeviceResources
 *
 * PARAMETERS:  ACPI_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Display the _PRT/_CRS/_PRS resources for a device object.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDbDeviceResources (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_NAMESPACE_NODE     *PrtNode = NULL;
    ACPI_NAMESPACE_NODE     *CrsNode = NULL;
    ACPI_NAMESPACE_NODE     *PrsNode = NULL;
    ACPI_NAMESPACE_NODE     *AeiNode = NULL;
    char                    *ParentPath;
    ACPI_BUFFER             ReturnBuffer;
    ACPI_STATUS             Status;


    Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
    ParentPath = AcpiNsGetExternalPathname (Node);
    if (!ParentPath)
    {
        return (AE_NO_MEMORY);
    }
```

```
919     /* Get handles to the resource methods for this device */

921     (void) AcpiGetHandle (Node, METHOD_NAME__PRT, ACPI_CAST_PTR (ACPI_HANDLE, &P
922     (void) AcpiGetHandle (Node, METHOD_NAME__CRS, ACPI_CAST_PTR (ACPI_HANDLE, &C
923     (void) AcpiGetHandle (Node, METHOD_NAME__PRS, ACPI_CAST_PTR (ACPI_HANDLE, &P
924     (void) AcpiGetHandle (Node, METHOD_NAME__AEI, ACPI_CAST_PTR (ACPI_HANDLE, &A
925     if (!PrtNode && !CrsNode && !PrsNode && !AeiNode)
926     {
927         goto Cleanup;   /* Nothing to do */
928     }

930     AcpiOsPrintf ("\nDevice: %s\n", ParentPath);

932     /* Prepare for a return object of arbitrary size */

934     ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
935     ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;


938     /* _PRT */

940     if (PrtNode)
941     {
942         AcpiOsPrintf ("Evaluating _PRT\n");

944         Status = AcpiEvaluateObject (PrtNode, NULL, NULL, &ReturnBuffer);
945         if (ACPI_FAILURE (Status))
946         {
947             AcpiOsPrintf ("Could not evaluate _PRT: %s\n",
948                 AcpiFormatException (Status));
949             goto GetCrs;
950         }

952         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
953         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

955         Status = AcpiGetIrqRoutingTable (Node, &ReturnBuffer);
956         if (ACPI_FAILURE (Status))
957         {
958             AcpiOsPrintf ("GetIrqRoutingTable failed: %s\n",
959                 AcpiFormatException (Status));
960             goto GetCrs;
961         }

963         AcpiRsDumpIrqList (ACPI_CAST_PTR (UINT8, AcpiGbl_DbBuffer));
964     }


967     /* _CRS */

969 GetCrs:
970     if (CrsNode)
971     {
972         AcpiOsPrintf ("Evaluating _CRS\n");

974         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
975         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

977         Status = AcpiEvaluateObject (CrsNode, NULL, NULL, &ReturnBuffer);
978         if (ACPI_FAILURE (Status))
979         {
980             AcpiOsPrintf ("Could not evaluate _CRS: %s\n",
981                 AcpiFormatException (Status));
982             goto GetPrs;
983         }
```



```
985         /* This code exercises the AcpiWalkResources interface */

987         Status = AcpiWalkResources (Node, METHOD_NAME__CRS,
988             AcpiDbResourceCallback, NULL);
989         if (ACPI_FAILURE (Status))
990         {
991             AcpiOsPrintf ("AcpiWalkResources failed: %s\n",
992                 AcpiFormatException (Status));
993             goto GetPrs;
994         }

996         /* Get the _CRS resource list (test ALLOCATE buffer) */

998         ReturnBuffer.Pointer = NULL;
999         ReturnBuffer.Length  = ACPI_ALLOCATE_LOCAL_BUFFER;

1001        Status = AcpiGetCurrentResources (Node, &ReturnBuffer);
1002        if (ACPI_FAILURE (Status))
1003        {
1004            AcpiOsPrintf ("AcpiGetCurrentResources failed: %s\n",
1005                AcpiFormatException (Status));
1006            goto GetPrs;
1007        }

1009        /* This code exercises the AcpiWalkResourceBuffer interface */

1011        Status = AcpiWalkResourceBuffer (&ReturnBuffer,
1012            AcpiDbResourceCallback, NULL);
1013        if (ACPI_FAILURE (Status))
1014        {
1015            AcpiOsPrintf ("AcpiWalkResourceBuffer failed: %s\n",
1016                AcpiFormatException (Status));
1017            goto EndCrs;
1018        }

1020        /* Dump the _CRS resource list */

1022        AcpiRsDumpResourceList (ACPI_CAST_PTR (ACPI_RESOURCE,
1023            ReturnBuffer.Pointer));

1025        /*
1026         * Perform comparison of original AML to newly created AML. This
1027         * tests both the AML->Resource conversion and the Resource->AML
1028         * conversion.
1029         */
1030        (void) AcpiDmTestResourceConversion (Node, METHOD_NAME__CRS);

1032        /* Execute _SRS with the resource list */

1034        AcpiOsPrintf ("Evaluating _SRS\n");

1036        Status = AcpiSetCurrentResources (Node, &ReturnBuffer);
1037        if (ACPI_FAILURE (Status))
1038        {
1039            AcpiOsPrintf ("AcpiSetCurrentResources failed: %s\n",
1040                AcpiFormatException (Status));
1041            goto EndCrs;
1042        }

1044 EndCrs:
1045        ACPI_FREE (ReturnBuffer.Pointer);
1046    }


1049    /* _PRS */
```

```
1051 GetPrs:
1052     if (PrsNode)
1053     {
1054         AcpiOsPrintf ("Evaluating _PRS\n");

1056         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
1057         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

1059         Status = AcpiEvaluateObject (PrsNode, NULL, NULL, &ReturnBuffer);
1060         if (ACPI_FAILURE (Status))
1061         {
1062             AcpiOsPrintf ("Could not evaluate _PRS: %s\n",
1063                 AcpiFormatException (Status));
1064             goto GetAei;
1065         }

1067         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
1068         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

1070         Status = AcpiGetPossibleResources (Node, &ReturnBuffer);
1071         if (ACPI_FAILURE (Status))
1072         {
1073             AcpiOsPrintf ("AcpiGetPossibleResources failed: %s\n",
1074                 AcpiFormatException (Status));
1075             goto GetAei;
1076         }

1078         AcpiRsDumpResourceList (ACPI_CAST_PTR (ACPI_RESOURCE, AcpiGbl_DbBuffer));
1079     }


1082     /* _AEI */

1084 GetAei:
1085     if (AeiNode)
1086     {
1087         AcpiOsPrintf ("Evaluating _AEI\n");

1089         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
1090         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

1092         Status = AcpiEvaluateObject (AeiNode, NULL, NULL, &ReturnBuffer);
1093         if (ACPI_FAILURE (Status))
1094         {
1095             AcpiOsPrintf ("Could not evaluate _AEI: %s\n",
1096                 AcpiFormatException (Status));
1097             goto Cleanup;
1098         }

1100         ReturnBuffer.Pointer = AcpiGbl_DbBuffer;
1101         ReturnBuffer.Length  = ACPI_DEBUG_BUFFER_SIZE;

1103         Status = AcpiGetEventResources (Node, &ReturnBuffer);
1104         if (ACPI_FAILURE (Status))
1105         {
1106             AcpiOsPrintf ("AcpiGetEventResources failed: %s\n",
1107                 AcpiFormatException (Status));
1108             goto Cleanup;
1109         }

1111         AcpiRsDumpResourceList (ACPI_CAST_PTR (ACPI_RESOURCE, AcpiGbl_DbBuffer));
1112     }


1115 Cleanup:
```



```
1116     ACPI_FREE (ParentPath);
1117     return (AE_OK);
1118 }


1121 /******************************************************************************
1122  *
1123  * FUNCTION:    AcpiDbDisplayResources
1124  *
1125  * PARAMETERS:  ObjectArg           - String object name or object pointer.
1126  *                                    NULL or "*" means "display resources for
1127  *                                    all devices"
1128  *
1129  * RETURN:      None
1130  *
1131  * DESCRIPTION: Display the resource objects associated with a device.
1132  *
1133  *****************************************************************************/

1135 void
1136 AcpiDbDisplayResources (
1137     char                    *ObjectArg)
1138 {
1139     ACPI_NAMESPACE_NODE     *Node;


1142     AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
1143     AcpiDbgLevel |= ACPI_LV_RESOURCES;

1145     /* Asterisk means "display resources for all devices" */

1147     if (!ObjectArg || (!ACPI_STRCMP (ObjectArg, "*")))
1148     {
1149         (void) AcpiWalkNamespace (ACPI_TYPE_DEVICE, ACPI_ROOT_OBJECT,
1150                     ACPI_UINT32_MAX, AcpiDbDeviceResources, NULL, NULL, NULL);
1151     }
1152     else
1153     {
1154         /* Convert string to object pointer */

1156         Node = AcpiDbConvertToNode (ObjectArg);
1157         if (Node)
1158         {
1159             if (Node->Type != ACPI_TYPE_DEVICE)
1160             {
1161                 AcpiOsPrintf ("%4.4s: Name is not a device object (%s)\n",
1162                     Node->Name.Ascii, AcpiUtGetTypeName (Node->Type));
1163             }
1164             else
1165             {
1166                 (void) AcpiDbDeviceResources (Node, 0, NULL, NULL);
1167             }
1168         }
1169     }

1171     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
1172 }


1175 #if (!ACPI_REDUCED_HARDWARE)
1176 /******************************************************************************
1177  *
1178  * FUNCTION:    AcpiDbGenerateGpe
1179  *
1180  * PARAMETERS:  GpeArg              - Raw GPE number, ascii string
1181  *              BlockArg            - GPE block number, ascii string
```

```
 *                                  0 or 1 for FADT GPE blocks
 *
 * RETURN:      None
 *
 * DESCRIPTION: Simulate firing of a GPE
 *
 *****************************************************************************/

void
AcpiDbGenerateGpe (
    char                    *GpeArg,
    char                    *BlockArg)
{
    UINT32                  BlockNumber;
    UINT32                  GpeNumber;
    ACPI_GPE_EVENT_INFO     *GpeEventInfo;


    GpeNumber   = ACPI_STRTOUL (GpeArg, NULL, 0);
    BlockNumber = ACPI_STRTOUL (BlockArg, NULL, 0);


    GpeEventInfo = AcpiEvGetGpeEventInfo (ACPI_TO_POINTER (BlockNumber),
        GpeNumber);
    if (!GpeEventInfo)
    {
        AcpiOsPrintf ("Invalid GPE\n");
        return;
    }

    (void) AcpiEvGpeDispatch (NULL, GpeEventInfo, GpeNumber);
}

void
AcpiDbGenerateSci (
    void)
{
    AcpiEvSciDispatch ();
}

#endif /* !ACPI_REDUCED_HARDWARE */

#endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   15819 Thu Dec 26 13:48:41 2013
new/usr/src/common/acpica/components/debugger/dbconvert.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbconvert - debugger miscellaneous conversion routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdebug.h"

#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbconvert")


#define DB_DEFAULT_PKG_ELEMENTS     33


/******************************************************************************
 *
 * FUNCTION:    AcpiDbHexCharToValue
 *
```



```
 * PARAMETERS:  HexChar             - Ascii Hex digit, 0-9|a-f|A-F
 *              ReturnValue         - Where the converted value is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert a single hex character to a 4-bit number (0-16).
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDbHexCharToValue (
    int                     HexChar,
    UINT8                   *ReturnValue)
{
    UINT8                   Value;


    /* Digit must be ascii [0-9a-fA-F] */

    if (!ACPI_IS_XDIGIT (HexChar))
    {
        return (AE_BAD_HEX_CONSTANT);
    }

    if (HexChar <= 0x39)
    {
        Value = (UINT8) (HexChar - 0x30);
    }
    else
    {
        Value = (UINT8) (ACPI_TOUPPER (HexChar) - 0x37);
    }

    *ReturnValue = Value;
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiDbHexByteToBinary
 *
 * PARAMETERS:  HexByte             - Double hex digit (0x00 - 0xFF) in format:
 *                                    HiByte then LoByte.
 *              ReturnValue         - Where the converted value is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert two hex characters to an 8 bit number (0 - 255).
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDbHexByteToBinary (
    char                    *HexByte,
    UINT8                   *ReturnValue)
{
    UINT8                   Local0;
    UINT8                   Local1;
    ACPI_STATUS             Status;


    /* High byte */

    Status = AcpiDbHexCharToValue (HexByte[0], &Local0);
    if (ACPI_FAILURE (Status))
```

```
128     {
129         return (Status);
130     }

132     /* Low byte */

134     Status = AcpiDbHexCharToValue (HexByte[1], &Local1);
135     if (ACPI_FAILURE (Status))
136     {
137         return (Status);
138     }

140     *ReturnValue = (UINT8) ((Local0 << 4) | Local1);
141     return (AE_OK);
142 }


145 /*******************************************************************************
146  *
147  * FUNCTION:    AcpiDbConvertToBuffer
148  *
149  * PARAMETERS:  String              - Input string to be converted
150  *              Object              - Where the buffer object is returned
151  *
152  * RETURN:      Status
153  *
154  * DESCRIPTION: Convert a string to a buffer object. String is treated a list
155  *              of buffer elements, each separated by a space or comma.
156  *
157  ******************************************************************************/

159 static ACPI_STATUS
160 AcpiDbConvertToBuffer (
161     char                    *String,
162     ACPI_OBJECT             *Object)
163 {
164     UINT32                  i;
165     UINT32                  j;
166     UINT32                  Length;
167     UINT8                   *Buffer;
168     ACPI_STATUS             Status;


171     /* Generate the final buffer length */

173     for (i = 0, Length = 0; String[i];)
174     {
175         i+=2;
176         Length++;

178         while (String[i] &&
179               ((String[i] == ',') || (String[i] == ' ')))
180         {
181             i++;
182         }
183     }

185     Buffer = ACPI_ALLOCATE (Length);
186     if (!Buffer)
187     {
188         return (AE_NO_MEMORY);
189     }

191     /* Convert the command line bytes to the buffer */

193     for (i = 0, j = 0; String[i];)
```



```
194     {
195         Status = AcpiDbHexByteToBinary (&String[i], &Buffer[j]);
196         if (ACPI_FAILURE (Status))
197         {
198             ACPI_FREE (Buffer);
199             return (Status);
200         }

202         j++;
203         i+=2;
204         while (String[i] &&
205               ((String[i] == ',') || (String[i] == ' ')))
206         {
207             i++;
208         }
209     }

211     Object->Type = ACPI_TYPE_BUFFER;
212     Object->Buffer.Pointer = Buffer;
213     Object->Buffer.Length = Length;
214     return (AE_OK);
215 }


218 /*******************************************************************************
219  *
220  * FUNCTION:    AcpiDbConvertToPackage
221  *
222  * PARAMETERS:  String              - Input string to be converted
223  *              Object              - Where the package object is returned
224  *
225  * RETURN:      Status
226  *
227  * DESCRIPTION: Convert a string to a package object. Handles nested packages
228  *              via recursion with AcpiDbConvertToObject.
229  *
230  ******************************************************************************/

232 ACPI_STATUS
233 AcpiDbConvertToPackage (
234     char                    *String,
235     ACPI_OBJECT             *Object)
236 {
237     char                    *This;
238     char                    *Next;
239     UINT32                  i;
240     ACPI_OBJECT_TYPE        Type;
241     ACPI_OBJECT             *Elements;
242     ACPI_STATUS             Status;


245     Elements = ACPI_ALLOCATE_ZEROED (
246         DB_DEFAULT_PKG_ELEMENTS * sizeof (ACPI_OBJECT));

248     This = String;
249     for (i = 0; i < (DB_DEFAULT_PKG_ELEMENTS - 1); i++)
250     {
251         This = AcpiDbGetNextToken (This, &Next, &Type);
252         if (!This)
253         {
254             break;
255         }

257         /* Recursive call to convert each package element */

259         Status = AcpiDbConvertToObject (Type, This, &Elements[i]);
```

```
260         if (ACPI_FAILURE (Status))
261         {
262             AcpiDbDeleteObjects (i + 1, Elements);
263             ACPI_FREE (Elements);
264             return (Status);
265         }
266
267         This = Next;
268     }
269
270     Object->Type = ACPI_TYPE_PACKAGE;
271     Object->Package.Count = i;
272     Object->Package.Elements = Elements;
273     return (AE_OK);
274 }
275
276
277 /*******************************************************************************
278  *
279  * FUNCTION:    AcpiDbConvertToObject
280  *
281  * PARAMETERS:  Type                - Object type as determined by parser
282  *              String              - Input string to be converted
283  *              Object              - Where the new object is returned
284  *
285  * RETURN:      Status
286  *
287  * DESCRIPTION: Convert a typed and tokenized string to an ACPI_OBJECT. Typing:
288  *              1) String objects were surrounded by quotes.
289  *              2) Buffer objects were surrounded by parentheses.
290  *              3) Package objects were surrounded by brackets "[]".
291  *              4) All standalone tokens are treated as integers.
292  *
293  ******************************************************************************/
294
295 ACPI_STATUS
296 AcpiDbConvertToObject (
297     ACPI_OBJECT_TYPE        Type,
298     char                    *String,
299     ACPI_OBJECT             *Object)
300 {
301     ACPI_STATUS             Status = AE_OK;
302
303
304     switch (Type)
305     {
306     case ACPI_TYPE_STRING:
307
308         Object->Type = ACPI_TYPE_STRING;
309         Object->String.Pointer = String;
310         Object->String.Length = (UINT32) ACPI_STRLEN (String);
311         break;
312
313     case ACPI_TYPE_BUFFER:
314
315         Status = AcpiDbConvertToBuffer (String, Object);
316         break;
317
318     case ACPI_TYPE_PACKAGE:
319
320         Status = AcpiDbConvertToPackage (String, Object);
321         break;
322
323     default:
324
325         Object->Type = ACPI_TYPE_INTEGER;
```



```
326         Status = AcpiUtStrtoul64 (String, 16, &Object->Integer.Value);
327         break;
328     }
329
330     return (Status);
331 }
332
333
334 /*******************************************************************************
335  *
336  * FUNCTION:    AcpiDbEncodePldBuffer
337  *
338  * PARAMETERS:  PldInfo             - _PLD buffer struct (Using local struct)
339  *
340  * RETURN:      Encode _PLD buffer suitable for return value from _PLD
341  *
342  * DESCRIPTION: Bit-packs a _PLD buffer struct. Used to test the _PLD macros
343  *
344  ******************************************************************************/
345
346 UINT8 *
347 AcpiDbEncodePldBuffer (
348     ACPI_PLD_INFO           *PldInfo)
349 {
350     UINT32                  *Buffer;
351     UINT32                  Dword;
352
353
354     Buffer = ACPI_ALLOCATE_ZEROED (ACPI_PLD_BUFFER_SIZE);
355     if (!Buffer)
356     {
357         return (NULL);
358     }
359
360     /* First 32 bits */
361
362     Dword = 0;
363     ACPI_PLD_SET_REVISION       (&Dword, PldInfo->Revision);
364     ACPI_PLD_SET_IGNORE_COLOR   (&Dword, PldInfo->IgnoreColor);
365     ACPI_PLD_SET_COLOR          (&Dword, PldInfo->Color);
366     ACPI_MOVE_32_TO_32 (&Buffer[0], &Dword);
367
368     /* Second 32 bits */
369
370     Dword = 0;
371     ACPI_PLD_SET_WIDTH          (&Dword, PldInfo->Width);
372     ACPI_PLD_SET_HEIGHT         (&Dword, PldInfo->Height);
373     ACPI_MOVE_32_TO_32 (&Buffer[1], &Dword);
374
375     /* Third 32 bits */
376
377     Dword = 0;
378     ACPI_PLD_SET_USER_VISIBLE   (&Dword, PldInfo->UserVisible);
379     ACPI_PLD_SET_DOCK           (&Dword, PldInfo->Dock);
380     ACPI_PLD_SET_LID            (&Dword, PldInfo->Lid);
381     ACPI_PLD_SET_PANEL          (&Dword, PldInfo->Panel);
382     ACPI_PLD_SET_VERTICAL       (&Dword, PldInfo->VerticalPosition);
383     ACPI_PLD_SET_HORIZONTAL     (&Dword, PldInfo->HorizontalPosition);
384     ACPI_PLD_SET_SHAPE          (&Dword, PldInfo->Shape);
385     ACPI_PLD_SET_ORIENTATION    (&Dword, PldInfo->GroupOrientation);
386     ACPI_PLD_SET_TOKEN          (&Dword, PldInfo->GroupToken);
387     ACPI_PLD_SET_POSITION       (&Dword, PldInfo->GroupPosition);
388     ACPI_PLD_SET_BAY            (&Dword, PldInfo->Bay);
389     ACPI_MOVE_32_TO_32 (&Buffer[2], &Dword);
390
391     /* Fourth 32 bits */
```

```
393     Dword = 0;
394     ACPI_PLD_SET_EJECTABLE      (&Dword, PldInfo->Ejectable);
395     ACPI_PLD_SET_OSPM_EJECT     (&Dword, PldInfo->OspmEjectRequired);
396     ACPI_PLD_SET_CABINET        (&Dword, PldInfo->CabinetNumber);
397     ACPI_PLD_SET_CARD_CAGE      (&Dword, PldInfo->CardCageNumber);
398     ACPI_PLD_SET_REFERENCE      (&Dword, PldInfo->Reference);
399     ACPI_PLD_SET_ROTATION       (&Dword, PldInfo->Rotation);
400     ACPI_PLD_SET_ORDER          (&Dword, PldInfo->Order);
401     ACPI_MOVE_32_TO_32 (&Buffer[3], &Dword);

403     if (PldInfo->Revision >= 2)
404     {
405         /* Fifth 32 bits */

407         Dword = 0;
408         ACPI_PLD_SET_VERT_OFFSET    (&Dword, PldInfo->VerticalOffset);
409         ACPI_PLD_SET_HORIZ_OFFSET   (&Dword, PldInfo->HorizontalOffset);
410         ACPI_MOVE_32_TO_32 (&Buffer[4], &Dword);
411     }

413     return (ACPI_CAST_PTR (UINT8, Buffer));
414 }


417 /*******************************************************************************
418  *
419  * FUNCTION:    AcpiDbDumpPldBuffer
420  *
421  * PARAMETERS:  ObjDesc             - Object returned from _PLD method
422  *
423  * RETURN:      None.
424  *
425  * DESCRIPTION: Dumps formatted contents of a _PLD return buffer.
426  *
427  ******************************************************************************/

429 #define ACPI_PLD_OUTPUT     "%20s : %-6X\n"

431 void
432 AcpiDbDumpPldBuffer (
433     ACPI_OBJECT             *ObjDesc)
434 {
435     ACPI_OBJECT             *BufferDesc;
436     ACPI_PLD_INFO           *PldInfo;
437     UINT8                   *NewBuffer;
438     ACPI_STATUS             Status;


441     /* Object must be of type Package with at least one Buffer element */

443     if (ObjDesc->Type != ACPI_TYPE_PACKAGE)
444     {
445         return;
446     }

448     BufferDesc = &ObjDesc->Package.Elements[0];
449     if (BufferDesc->Type != ACPI_TYPE_BUFFER)
450     {
451         return;
452     }

454     /* Convert _PLD buffer to local _PLD struct */

456     Status = AcpiDecodePldBuffer (BufferDesc->Buffer.Pointer,
457         BufferDesc->Buffer.Length, &PldInfo);
```



```
458     if (ACPI_FAILURE (Status))
459     {
460         return;
461     }

463     /* Encode local _PLD struct back to a _PLD buffer */

465     NewBuffer = AcpiDbEncodePldBuffer (PldInfo);
466     if (!NewBuffer)
467     {
468         return;
469     }

471     /* The two bit-packed buffers should match */

473     if (ACPI_MEMCMP (NewBuffer, BufferDesc->Buffer.Pointer,
474         BufferDesc->Buffer.Length))
475     {
476         AcpiOsPrintf ("Converted _PLD buffer does not compare. New:\n");

478         AcpiUtDumpBuffer (NewBuffer,
479             BufferDesc->Buffer.Length, DB_BYTE_DISPLAY, 0);
480     }

482     /* First 32-bit dword */

484     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Revision", PldInfo->Revision);
485     AcpiOsPrintf (ACPI_PLD_OUTPUT, "IgnoreColor", PldInfo->IgnoreColor);
486     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Color", PldInfo->Color);

488     /* Second 32-bit dword */

490     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Width", PldInfo->Width);
491     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Height", PldInfo->Height);

493     /* Third 32-bit dword */

495     AcpiOsPrintf (ACPI_PLD_OUTPUT, "UserVisible", PldInfo->UserVisible);
496     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Dock", PldInfo->Dock);
497     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Lid", PldInfo->Lid);
498     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Panel", PldInfo->Panel);
499     AcpiOsPrintf (ACPI_PLD_OUTPUT, "VerticalPosition", PldInfo->VerticalPosition
500     AcpiOsPrintf (ACPI_PLD_OUTPUT, "HorizontalPosition", PldInfo->HorizontalPosi
501     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Shape", PldInfo->Shape);
502     AcpiOsPrintf (ACPI_PLD_OUTPUT, "GroupOrientation", PldInfo->GroupOrientation
503     AcpiOsPrintf (ACPI_PLD_OUTPUT, "GroupToken", PldInfo->GroupToken);
504     AcpiOsPrintf (ACPI_PLD_OUTPUT, "GroupPosition", PldInfo->GroupPosition);
505     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Bay", PldInfo->Bay);

507     /* Fourth 32-bit dword */

509     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Ejectable", PldInfo->Ejectable);
510     AcpiOsPrintf (ACPI_PLD_OUTPUT, "OspmEjectRequired", PldInfo->OspmEjectRequir
511     AcpiOsPrintf (ACPI_PLD_OUTPUT, "CabinetNumber", PldInfo->CabinetNumber);
512     AcpiOsPrintf (ACPI_PLD_OUTPUT, "CardCageNumber", PldInfo->CardCageNumber);
513     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Reference", PldInfo->Reference);
514     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Rotation", PldInfo->Rotation);
515     AcpiOsPrintf (ACPI_PLD_OUTPUT, "Order", PldInfo->Order);

517     /* Fifth 32-bit dword */

519     if (BufferDesc->Buffer.Length > 16)
520     {
521         AcpiOsPrintf (ACPI_PLD_OUTPUT, "VerticalOffset", PldInfo->VerticalOffset
522         AcpiOsPrintf (ACPI_PLD_OUTPUT, "HorizontalOffset", PldInfo->HorizontalOf
523     }
```

```
525     ACPI_FREE (PldInfo);
526     ACPI_FREE (NewBuffer);
527 }

529 #endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   32871 Thu Dec 26 13:48:41 2013
new/usr/src/common/acpica/components/debugger/dbdisply.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbdisply - debug display commands
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acdispat.h"
#include "acnamesp.h"
#include "acparser.h"
#include "acinterp.h"
#include "acdebug.h"
#include "acdisasm.h"


#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbdisply")
```



```
/* Local prototypes */

static void
AcpiDbDumpParserDescriptor (
    ACPI_PARSE_OBJECT       *Op);

static void *
AcpiDbGetPointer (
    void                    *Target);

static ACPI_STATUS
AcpiDbDisplayNonRootHandlers (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);

/*
 * System handler information.
 * Used for Handlers command, in AcpiDbDisplayHandlers.
 */
#define ACPI_PREDEFINED_PREFIX          "%25s (%.2X) : "
#define ACPI_HANDLER_NAME_STRING                "%30s : "
#define ACPI_HANDLER_PRESENT_STRING                     "%-9s (%p)\n"
#define ACPI_HANDLER_PRESENT_STRING2                    "%-9s (%p)"
#define ACPI_HANDLER_NOT_PRESENT_STRING                 "%-9s\n"

/* All predefined Address Space IDs */

static ACPI_ADR_SPACE_TYPE  AcpiGbl_SpaceIdList[] =
{
    ACPI_ADR_SPACE_SYSTEM_MEMORY,
    ACPI_ADR_SPACE_SYSTEM_IO,
    ACPI_ADR_SPACE_PCI_CONFIG,
    ACPI_ADR_SPACE_EC,
    ACPI_ADR_SPACE_SMBUS,
    ACPI_ADR_SPACE_CMOS,
    ACPI_ADR_SPACE_PCI_BAR_TARGET,
    ACPI_ADR_SPACE_IPMI,
    ACPI_ADR_SPACE_GPIO,
    ACPI_ADR_SPACE_GSBUS,
    ACPI_ADR_SPACE_DATA_TABLE,
    ACPI_ADR_SPACE_FIXED_HARDWARE
};
```

*unchanged_portion_omitted_*

```
static ACPI_HANDLER_INFO    AcpiGbl_HandlerList[] =
{
    {&AcpiGbl_GlobalNotify[0].Handler,  "System Notifications"},
    {&AcpiGbl_GlobalNotify[1].Handler,  "Device Notifications"},
    {&AcpiGbl_SystemNotify.Handler,     "System Notifications"},
    {&AcpiGbl_DeviceNotify.Handler,     "Device Notifications"},
    {&AcpiGbl_TableHandler,             "ACPI Table Events"},
    {&AcpiGbl_ExceptionHandler,         "Control Method Exceptions"},
    {&AcpiGbl_InterfaceHandler,         "OSI Invocations"}
};


/******************************************************************************
 *
 * FUNCTION:    AcpiDbGetPointer
 *
 * PARAMETERS:  Target          - Pointer to string to be converted
 *
 * RETURN:      Converted pointer
```

```
 132  *
 133  * DESCRIPTION: Convert an ascii pointer value to a real value
 134  *
 135  ******************************************************************************/

 137 static void *
 138 AcpiDbGetPointer (
 139     void                    *Target)
 140 {
 141     void                    *ObjPtr;
 142     ACPI_SIZE               Address;


 145     Address = ACPI_STRTOUL (Target, NULL, 16);
 146     ObjPtr = ACPI_TO_POINTER (Address);
 135     ObjPtr = ACPI_TO_POINTER (ACPI_STRTOUL (Target, NULL, 16));
 147     return (ObjPtr);
 148 }
_____unchanged_portion_omitted_


 184 /******************************************************************************
 185  *
 186  * FUNCTION:    AcpiDbDecodeAndDisplayObject
 187  *
 188  * PARAMETERS:  Target          - String with object to be displayed. Names
 189  *                                and hex pointers are supported.
 190  *              OutputType      - Byte, Word, Dword, or Qword (B|W|D|Q)
 191  *
 192  * RETURN:      None
 193  *
 194  * DESCRIPTION: Display a formatted ACPI object
 195  *
 196  ******************************************************************************/

 198 void
 199 AcpiDbDecodeAndDisplayObject (
 200     char                    *Target,
 201     char                    *OutputType)
 202 {
 203     void                    *ObjPtr;
 204     ACPI_NAMESPACE_NODE     *Node;
 205     ACPI_OPERAND_OBJECT     *ObjDesc;
 206     UINT32                  Display = DB_BYTE_DISPLAY;
 207     char                    Buffer[80];
 208     ACPI_BUFFER             RetBuf;
 209     ACPI_STATUS             Status;
 210     UINT32                  Size;


 213     if (!Target)
 214     {
 215         return;
 216     }

 218     /* Decode the output type */

 220     if (OutputType)
 221     {
 222         AcpiUtStrupr (OutputType);
 223         if (OutputType[0] == 'W')
 224         {
 225             Display = DB_WORD_DISPLAY;
 226         }
 227         else if (OutputType[0] == 'D')
 228         {
```



```
 229             Display = DB_DWORD_DISPLAY;
 230         }
 231         else if (OutputType[0] == 'Q')
 232         {
 233             Display = DB_QWORD_DISPLAY;
 234         }
 235     }

 237     RetBuf.Length = sizeof (Buffer);
 238     RetBuf.Pointer = Buffer;

 240     /* Differentiate between a number and a name */

 242     if ((Target[0] >= 0x30) && (Target[0] <= 0x39))
 243     {
 244         ObjPtr = AcpiDbGetPointer (Target);
 245         if (!AcpiOsReadable (ObjPtr, 16))
 246         {
 247             AcpiOsPrintf ("Address %p is invalid in this address space\n",
 248                 ObjPtr);
 249             return;
 250         }

 252         /* Decode the object type */

 254         switch (ACPI_GET_DESCRIPTOR_TYPE (ObjPtr))
 255         {
 256         case ACPI_DESC_TYPE_NAMED:

 258             /* This is a namespace Node */

 260             if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_NAMESPACE_NODE)))
 261             {
 262                 AcpiOsPrintf (
 263                     "Cannot read entire Named object at address %p\n", ObjPtr);
 264                 return;
 265             }

 267             Node = ObjPtr;
 268             goto DumpNode;


 270         case ACPI_DESC_TYPE_OPERAND:

 272             /* This is a ACPI OPERAND OBJECT */

 274             if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_OPERAND_OBJECT)))
 275             {
 276                 AcpiOsPrintf ("Cannot read entire ACPI object at address %p\n",
 277                     ObjPtr);
 278                 return;
 279             }

 281             AcpiUtDebugDumpBuffer (ObjPtr, sizeof (ACPI_OPERAND_OBJECT), Display
 271             AcpiUtDumpBuffer (ObjPtr, sizeof (ACPI_OPERAND_OBJECT), Display,
 282                 ACPI_UINT32_MAX);
 283             AcpiExDumpObjectDescriptor (ObjPtr, 1);
 284             break;


 286         case ACPI_DESC_TYPE_PARSER:

 288             /* This is a Parser Op object */

 290             if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_PARSE_OBJECT)))
 291             {
```

```
292                 AcpiOsPrintf (
293                     "Cannot read entire Parser object at address %p\n", ObjPtr);
294                 return;
295             }

297             AcpiUtDebugDumpBuffer (ObjPtr, sizeof (ACPI_PARSE_OBJECT), Display,
288             AcpiUtDumpBuffer (ObjPtr, sizeof (ACPI_PARSE_OBJECT), Display,
298                 ACPI_UINT32_MAX);
299             AcpiDbDumpParserDescriptor ((ACPI_PARSE_OBJECT *) ObjPtr);
300             break;


302         default:

304             /* Is not a recognizeable object */

306             Size = 16;
307             if (AcpiOsReadable (ObjPtr, 64))
308             {
309                 Size = 64;
310             }

312             /* Just dump some memory */

314             AcpiUtDebugDumpBuffer (ObjPtr, Size, Display, ACPI_UINT32_MAX);
306             AcpiUtDumpBuffer (ObjPtr, Size, Display, ACPI_UINT32_MAX);
315             break;
316         }

318         return;
319     }

321     /* The parameter is a name string that must be resolved to a Named obj */

323     Node = AcpiDbLocalNsLookup (Target);
324     if (!Node)
325     {
326         return;
327     }


330 DumpNode:
331     /* Now dump the NS node */

333     Status = AcpiGetName (Node, ACPI_FULL_PATHNAME, &RetBuf);
334     if (ACPI_FAILURE (Status))
335     {
336         AcpiOsPrintf ("Could not convert name to pathname\n");
337     }

339     else
340     {
341         AcpiOsPrintf ("Object (%p) Pathname:  %s\n",
342             Node, (char *) RetBuf.Pointer);
343     }

345     if (!AcpiOsReadable (Node, sizeof (ACPI_NAMESPACE_NODE)))
346     {
347         AcpiOsPrintf ("Invalid Named object at address %p\n", Node);
348         return;
349     }

351     AcpiUtDebugDumpBuffer ((void *) Node, sizeof (ACPI_NAMESPACE_NODE),
343     AcpiUtDumpBuffer ((void *) Node, sizeof (ACPI_NAMESPACE_NODE),
352         Display, ACPI_UINT32_MAX);
353     AcpiExDumpNamespaceNode (Node, 1);
```



```
355     ObjDesc = AcpiNsGetAttachedObject (Node);
356     if (ObjDesc)
357     {
358         AcpiOsPrintf ("\nAttached Object (%p):\n", ObjDesc);
359         if (!AcpiOsReadable (ObjDesc, sizeof (ACPI_OPERAND_OBJECT)))
360         {
361             AcpiOsPrintf ("Invalid internal ACPI Object at address %p\n",
362                 ObjDesc);
363             return;
364         }

366         AcpiUtDebugDumpBuffer ((void *) ObjDesc, sizeof (ACPI_OPERAND_OBJECT),
358         AcpiUtDumpBuffer ((void *) ObjDesc, sizeof (ACPI_OPERAND_OBJECT),
367             Display, ACPI_UINT32_MAX);
368         AcpiExDumpObjectDescriptor (ObjDesc, 1);
369     }
370 }


373 /*******************************************************************************
374  *
375  * FUNCTION:    AcpiDbDisplayMethodInfo
376  *
377  * PARAMETERS:  StartOp         - Root of the control method parse tree
378  *
379  * RETURN:      None
380  *
381  * DESCRIPTION: Display information about the current method
382  *
383  ******************************************************************************/

385 void
386 AcpiDbDisplayMethodInfo (
387     ACPI_PARSE_OBJECT       *StartOp)
388 {
389     ACPI_WALK_STATE         *WalkState;
390     ACPI_OPERAND_OBJECT     *ObjDesc;
391     ACPI_NAMESPACE_NODE     *Node;
392     ACPI_PARSE_OBJECT       *RootOp;
393     ACPI_PARSE_OBJECT       *Op;
394     const ACPI_OPCODE_INFO  *OpInfo;
395     UINT32                  NumOps = 0;
396     UINT32                  NumOperands = 0;
397     UINT32                  NumOperators = 0;
398     UINT32                  NumRemainingOps = 0;
399     UINT32                  NumRemainingOperands = 0;
400     UINT32                  NumRemainingOperators = 0;
401     BOOLEAN                 CountRemaining = FALSE;


404     WalkState = AcpiDsGetCurrentWalkState (AcpiGbl_CurrentWalkList);
405     if (!WalkState)
406     {
407         AcpiOsPrintf ("There is no method currently executing\n");
408         return;
409     }

411     ObjDesc = WalkState->MethodDesc;
412     Node    = WalkState->MethodNode;

414     AcpiOsPrintf ("Currently executing control method is [%4.4s]\n",
415             AcpiUtGetNodeName (Node));
416     AcpiOsPrintf ("%X Arguments, SyncLevel = %X\n",
417             (UINT32) ObjDesc->Method.ParamCount,
418             (UINT32) ObjDesc->Method.SyncLevel);
```

```
421     RootOp = StartOp;
422     while (RootOp->Common.Parent)
423     {
424         RootOp = RootOp->Common.Parent;
425     }

427     Op = RootOp;

429     while (Op)
430     {
431         if (Op == StartOp)
432         {
433             CountRemaining = TRUE;
434         }

436         NumOps++;
437         if (CountRemaining)
438         {
439             NumRemainingOps++;
440         }

442         /* Decode the opcode */

444         OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
445         switch (OpInfo->Class)
446         {
447         case AML_CLASS_ARGUMENT:

449             if (CountRemaining)
450             {
451                 NumRemainingOperands++;
452             }

454             NumOperands++;
455             break;

457         case AML_CLASS_UNKNOWN:

459             /* Bad opcode or ASCII character */

461             continue;

463         default:

465             if (CountRemaining)
466             {
467                 NumRemainingOperators++;
468             }

470             NumOperators++;
471             break;
472         }

474         Op = AcpiPsGetDepthNext (StartOp, Op);
475     }

477     AcpiOsPrintf (
478         "Method contains:       %X AML Opcodes - %X Operators, %X Operands\n",
479         NumOps, NumOperators, NumOperands);

481     AcpiOsPrintf (
482         "Remaining to execute:  %X AML Opcodes - %X Operators, %X Operands\n",
483         NumRemainingOps, NumRemainingOperators, NumRemainingOperands);
484 }
```

*_____unchanged_portion_omitted_*


```
651 /*******************************************************************************
652  *
653  * FUNCTION:    AcpiDbDisplayObjectType
654  *
655  * PARAMETERS:  Name            - User entered NS node handle or name
644  * PARAMETERS:  ObjectArg       - User entered NS node handle
656  *
657  * RETURN:      None
658  *
659  * DESCRIPTION: Display type of an arbitrary NS node
660  *
661  ******************************************************************************/

663 void
664 AcpiDbDisplayObjectType (
665     char                    *Name)
654     char                    *ObjectArg)
666 {
667     ACPI_NAMESPACE_NODE     *Node;
656     ACPI_HANDLE             Handle;
668     ACPI_DEVICE_INFO        *Info;
669     ACPI_STATUS             Status;
670     UINT32                  i;


673     Node = AcpiDbConvertToNode (Name);
674     if (!Node)
675     {
676         return;
677     }
662     Handle = ACPI_TO_POINTER (ACPI_STRTOUL (ObjectArg, NULL, 16));

679     Status = AcpiGetObjectInfo (ACPI_CAST_PTR (ACPI_HANDLE, Node), &Info);
664     Status = AcpiGetObjectInfo (Handle, &Info);
680     if (ACPI_FAILURE (Status))
681     {
682         AcpiOsPrintf ("Could not get object info, %s\n",
683             AcpiFormatException (Status));
684         return;
685     }

687     if (Info->Valid & ACPI_VALID_ADR)
688     {
689         AcpiOsPrintf ("ADR: %8.8X%8.8X, STA: %8.8X, Flags: %X\n",
690             ACPI_FORMAT_UINT64 (Info->Address),
691             Info->CurrentStatus, Info->Flags);
692     }
693     if (Info->Valid & ACPI_VALID_SXDS)
694     {

695         AcpiOsPrintf ("S1D-%2.2X S2D-%2.2X S3D-%2.2X S4D-%2.2X\n",
696             Info->HighestDstates[0], Info->HighestDstates[1],
697             Info->HighestDstates[2], Info->HighestDstates[3]);
698     }
699     if (Info->Valid & ACPI_VALID_SXWS)
700     {

701         AcpiOsPrintf ("S0W-%2.2X S1W-%2.2X S2W-%2.2X S3W-%2.2X S4W-%2.2X\n",
702             Info->LowestDstates[0], Info->LowestDstates[1],
703             Info->LowestDstates[2], Info->LowestDstates[3],
704             Info->LowestDstates[4]);
705     }

707     if (Info->Valid & ACPI_VALID_HID)
```

```
    {
        AcpiOsPrintf ("HID: %s\n", Info->HardwareId.String);
    }
    if (Info->Valid & ACPI_VALID_UID)
    {
        AcpiOsPrintf ("UID: %s\n", Info->UniqueId.String);
    }
    if (Info->Valid & ACPI_VALID_SUB)
    {
        AcpiOsPrintf ("SUB: %s\n", Info->SubsystemId.String);
    }
    if (Info->Valid & ACPI_VALID_CID)
    {
        for (i = 0; i < Info->CompatibleIdList.Count; i++)
        {
            AcpiOsPrintf ("CID %u: %s\n", i,
                Info->CompatibleIdList.Ids[i].String);
        }
    }

    ACPI_FREE (Info);
}
```

*unchanged_portion_omitted_*

```
#if (!ACPI_REDUCED_HARDWARE)
/******************************************************************************
 *
 * FUNCTION:    AcpiDbDisplayGpes
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display the current GPE structures
 *
 *****************************************************************************/

void
AcpiDbDisplayGpes (
    void)
{
    ACPI_GPE_BLOCK_INFO     *GpeBlock;
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo;
    ACPI_GPE_EVENT_INFO     *GpeEventInfo;
    ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
    char                    *GpeType;
    ACPI_GPE_NOTIFY_INFO    *Notify;
    UINT32                  GpeIndex;
    UINT32                  Block = 0;
    UINT32                  i;
    UINT32                  j;
    UINT32                  Count;
    char                    Buffer[80];
    ACPI_BUFFER             RetBuf;
    ACPI_STATUS             Status;


    RetBuf.Length = sizeof (Buffer);
    RetBuf.Pointer = Buffer;

    Block = 0;

    /* Walk the GPE lists */

    GpeXruptInfo = AcpiGbl_GpeXruptListHead;
```



```
    while (GpeXruptInfo)
    {
        GpeBlock = GpeXruptInfo->GpeBlockListHead;
        while (GpeBlock)
        {
            Status = AcpiGetName (GpeBlock->Node, ACPI_FULL_PATHNAME, &RetBuf);
            if (ACPI_FAILURE (Status))
            {
                AcpiOsPrintf ("Could not convert name to pathname\n");
            }

            if (GpeBlock->Node == AcpiGbl_FadtGpeDevice)
            {
                GpeType = "FADT-defined GPE block";
            }
            else
            {
                GpeType = "GPE Block Device";
            }

            AcpiOsPrintf ("\nBlock %u - Info %p  DeviceNode %p [%s] - %s\n",
                Block, GpeBlock, GpeBlock->Node, Buffer, GpeType);

            AcpiOsPrintf ("    Registers:    %u (%u GPEs)\n",
                GpeBlock->RegisterCount, GpeBlock->GpeCount);

            AcpiOsPrintf ("    GPE range:    0x%X to 0x%X on interrupt %u\n",
                GpeBlock->BlockBaseNumber,
                GpeBlock->BlockBaseNumber + (GpeBlock->GpeCount - 1),
                GpeXruptInfo->InterruptNumber);

            AcpiOsPrintf (
                "    RegisterInfo: %p  Status %8.8X%8.8X Enable %8.8X%8.8X\n",
                GpeBlock->RegisterInfo,
                ACPI_FORMAT_UINT64 (GpeBlock->RegisterInfo->StatusAddress.Addres
                ACPI_FORMAT_UINT64 (GpeBlock->RegisterInfo->EnableAddress.Addres

            AcpiOsPrintf ("    EventInfo:    %p\n", GpeBlock->EventInfo);

            /* Examine each GPE Register within the block */

            for (i = 0; i < GpeBlock->RegisterCount; i++)
            {
                GpeRegisterInfo = &GpeBlock->RegisterInfo[i];

                AcpiOsPrintf (
                    "    Reg %u: (GPE %.2X-%.2X)  RunEnable %2.2X WakeEnable %2.
                    " Status %8.8X%8.8X Enable %8.8X%8.8X\n",
                    i, GpeRegisterInfo->BaseGpeNumber,
                    GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH -
                    GpeRegisterInfo->EnableForRun,
                    GpeRegisterInfo->EnableForWake,
                    ACPI_FORMAT_UINT64 (GpeRegisterInfo->StatusAddress.Address),
                    ACPI_FORMAT_UINT64 (GpeRegisterInfo->EnableAddress.Address))

                /* Now look at the individual GPEs in this byte register */

                for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
                {
                    GpeIndex = (i * ACPI_GPE_REGISTER_WIDTH) + j;
                    GpeEventInfo = &GpeBlock->EventInfo[GpeIndex];

                    if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
                        ACPI_GPE_DISPATCH_NONE)
                    {
                        /* This GPE is not used (no method or handler), ignore i
```

```
905                         continue;
906                     }
907
908                     AcpiOsPrintf (
909                         "        GPE %.2X: %p  RunRefs %2.2X Flags %2.2X (",
910                         GpeBlock->BlockBaseNumber + GpeIndex, GpeEventInfo,
911                         GpeEventInfo->RuntimeCount, GpeEventInfo->Flags);
912
913                     /* Decode the flags byte */
914
915                     if (GpeEventInfo->Flags & ACPI_GPE_LEVEL_TRIGGERED)
916                     {
917                         AcpiOsPrintf ("Level, ");
918                     }
919                     else
920                     {
921                         AcpiOsPrintf ("Edge,  ");
922                     }
923
924                     if (GpeEventInfo->Flags & ACPI_GPE_CAN_WAKE)
925                     {
926                         AcpiOsPrintf ("CanWake, ");
927                     }
928                     else
929                     {
930                         AcpiOsPrintf ("RunOnly, ");
931                     }
932
933                     switch (GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK)
934                     {
935                     case ACPI_GPE_DISPATCH_NONE:
936
937                         AcpiOsPrintf ("NotUsed");
938                         break;
939
940                     case ACPI_GPE_DISPATCH_METHOD:
941
942                         AcpiOsPrintf ("Method");
943                         break;
944                     case ACPI_GPE_DISPATCH_HANDLER:
945
946                         AcpiOsPrintf ("Handler");
947                         break;
948
949                     case ACPI_GPE_DISPATCH_NOTIFY:

951                         Count = 0;
952                         Notify = GpeEventInfo->Dispatch.NotifyList;
953                         while (Notify)
954                         {
955                             Count++;
956                             Notify = Notify->Next;
957                         }
958                         AcpiOsPrintf ("Implicit Notify on %u devices", Count);
959                         AcpiOsPrintf ("Notify");
960                         break;
961                     default:
962
963                         AcpiOsPrintf ("UNKNOWN: %X",
964                             GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK);
965                         break;
966                     }
967
968                     AcpiOsPrintf (")\n");
```



```
969                 }
970             }
971             Block++;
972             GpeBlock = GpeBlock->Next;
973         }
974         GpeXruptInfo = GpeXruptInfo->Next;
975     }
976 }
977 #endif /* !ACPI_REDUCED_HARDWARE */
978
979
980 /*******************************************************************************
981  *
982  * FUNCTION:    AcpiDbDisplayHandlers
983  *
984  * PARAMETERS:  None
985  *
986  * RETURN:      None
987  *
988  * DESCRIPTION: Display the currently installed global handlers
989  *
990  ******************************************************************************/
991
992 void
993 AcpiDbDisplayHandlers (
994     void)
995 {
996     ACPI_OPERAND_OBJECT     *ObjDesc;
997     ACPI_OPERAND_OBJECT     *HandlerObj;
998     ACPI_ADR_SPACE_TYPE     SpaceId;
999     UINT32                  i;
1000
1001
1002     /* Operation region handlers */
1003
1004     AcpiOsPrintf ("\nOperation Region Handlers at the namespace root:\n");
959      AcpiOsPrintf ("\nOperation Region Handlers:\n");
1005
1006     ObjDesc = AcpiNsGetAttachedObject (AcpiGbl_RootNode);
1007     if (ObjDesc)
1008     {
1009         for (i = 0; i < ACPI_ARRAY_LENGTH (AcpiGbl_SpaceIdList); i++)
1010         {
1011             SpaceId = AcpiGbl_SpaceIdList[i];
1012             HandlerObj = ObjDesc->Device.Handler;
1013
1014             AcpiOsPrintf (ACPI_PREDEFINED_PREFIX,
1015                 AcpiUtGetRegionName ((UINT8) SpaceId), SpaceId);
1016
1017             while (HandlerObj)
1018             {
1019                 if (AcpiGbl_SpaceIdList[i] == HandlerObj->AddressSpace.SpaceId)
974                  if (i == HandlerObj->AddressSpace.SpaceId)
1020                 {
1021                     AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING,
1022                         (HandlerObj->AddressSpace.HandlerFlags &
1023                             ACPI_ADDR_HANDLER_DEFAULT_INSTALLED) ? "Default" : "
1024                         HandlerObj->AddressSpace.Handler);
1025                     goto FoundHandler;
1026                 }
1027
1028                 HandlerObj = HandlerObj->AddressSpace.Next;
1029             }
1030
1031             /* There is no handler for this SpaceId */
```

```
1033             AcpiOsPrintf ("None\n");
1035         FoundHandler:;
1036         }
1038         /* Find all handlers for user-defined SpaceIDs */
1040         HandlerObj = ObjDesc->Device.Handler;
1041         while (HandlerObj)
1042         {
1043             if (HandlerObj->AddressSpace.SpaceId >= ACPI_USER_REGION_BEGIN)
1044             {
1045                 AcpiOsPrintf (ACPI_PREDEFINED_PREFIX,
1046                     "User-defined ID", HandlerObj->AddressSpace.SpaceId);
1047                 AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING,
1048                     (HandlerObj->AddressSpace.HandlerFlags &
1049                         ACPI_ADDR_HANDLER_DEFAULT_INSTALLED) ? "Default" : "User
1050                     HandlerObj->AddressSpace.Handler);
1051             }
1053             HandlerObj = HandlerObj->AddressSpace.Next;
1054         }
1055     }
1057 #if (!ACPI_REDUCED_HARDWARE)
1059     /* Fixed event handlers */
1061     AcpiOsPrintf ("\nFixed Event Handlers:\n");
1063     for (i = 0; i < ACPI_NUM_FIXED_EVENTS; i++)
1064     {
1065         AcpiOsPrintf (ACPI_PREDEFINED_PREFIX, AcpiUtGetEventName (i), i);
1066         if (AcpiGbl_FixedEventHandlers[i].Handler)
1067         {
1068             AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING, "User",
1069                 AcpiGbl_FixedEventHandlers[i].Handler);
1070         }
1071         else
1072         {
1073             AcpiOsPrintf (ACPI_HANDLER_NOT_PRESENT_STRING, "None");
1074         }
1075     }
1077 #endif /* !ACPI_REDUCED_HARDWARE */
1079     /* Miscellaneous global handlers */
1081     AcpiOsPrintf ("\nMiscellaneous Global Handlers:\n");
1083     for (i = 0; i < ACPI_ARRAY_LENGTH (AcpiGbl_HandlerList); i++)
1084     {
1085         AcpiOsPrintf (ACPI_HANDLER_NAME_STRING, AcpiGbl_HandlerList[i].Name);
1086         if (AcpiGbl_HandlerList[i].Handler)
1087         {
1088             AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING, "User",
1089                 AcpiGbl_HandlerList[i].Handler);
1090         }
1091         else
1092         {
1093             AcpiOsPrintf (ACPI_HANDLER_NOT_PRESENT_STRING, "None");
1094         }
1095     }
1098     /* Other handlers that are installed throughout the namespace */
```



```
1100     AcpiOsPrintf ("\nOperation Region Handlers for specific devices:\n");
1102     (void) AcpiWalkNamespace (ACPI_TYPE_DEVICE, ACPI_ROOT_OBJECT,
1103                 ACPI_UINT32_MAX, AcpiDbDisplayNonRootHandlers,
1104                 NULL, NULL, NULL);
1105 }
1108 /*******************************************************************************
1109  *
1110  * FUNCTION:    AcpiDbDisplayNonRootHandlers
1111  *
1112  * PARAMETERS:  ACPI_WALK_CALLBACK
1113  *
1114  * RETURN:      Status
1115  *
1116  * DESCRIPTION: Display information about all handlers installed for a
1117  *              device object.
1118  *
1119  ******************************************************************************/
1121 static ACPI_STATUS
1122 AcpiDbDisplayNonRootHandlers (
1123     ACPI_HANDLE             ObjHandle,
1124     UINT32                  NestingLevel,
1125     void                    *Context,
1126     void                    **ReturnValue)
1127 {
1128     ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandl
1129     ACPI_OPERAND_OBJECT     *ObjDesc;
1130     ACPI_OPERAND_OBJECT     *HandlerObj;
1131     char                    *Pathname;
1134     ObjDesc = AcpiNsGetAttachedObject (Node);
1135     if (!ObjDesc)
1136     {
1137         return (AE_OK);
1138     }
1140     Pathname = AcpiNsGetExternalPathname (Node);
1141     if (!Pathname)
1142     {
1143         return (AE_OK);
1144     }
1146     /* Display all handlers associated with this device */
1148     HandlerObj = ObjDesc->Device.Handler;
1149     while (HandlerObj)
1150     {
1151         AcpiOsPrintf (ACPI_PREDEFINED_PREFIX,
1152             AcpiUtGetRegionName ((UINT8) HandlerObj->AddressSpace.SpaceId),
1153             HandlerObj->AddressSpace.SpaceId);
1155         AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING2,
1156             (HandlerObj->AddressSpace.HandlerFlags &
1157                 ACPI_ADDR_HANDLER_DEFAULT_INSTALLED) ? "Default" : "User",
1158             HandlerObj->AddressSpace.Handler);
1160         AcpiOsPrintf (" Device Name: %s (%p)\n", Pathname, Node);
1162         HandlerObj = HandlerObj->AddressSpace.Next;
1163     }
```

```
1165     ACPI_FREE (Pathname);
1166     return (AE_OK);
1167 }

1169 #endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   23658 Thu Dec 26 13:48:41 2013
new/usr/src/common/acpica/components/debugger/dbexec.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbexec - debugger control method execution
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdebug.h"
#include "acnamesp.h"

#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbexec")


static ACPI_DB_METHOD_INFO          AcpiGbl_DbMethodInfo;

/* Local prototypes */
```



```
static ACPI_STATUS
AcpiDbExecuteMethod (
    ACPI_DB_METHOD_INFO     *Info,
    ACPI_BUFFER             *ReturnObj);

static ACPI_STATUS
AcpiDbExecuteSetup (
    ACPI_DB_METHOD_INFO     *Info);

static UINT32
AcpiDbGetOutstandingAllocations (
    void);

static void ACPI_SYSTEM_XFACE
AcpiDbMethodThread (
    void                    *Context);

static ACPI_STATUS
AcpiDbExecutionWalk (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    AcpiDbDeleteObjects
 *
 * PARAMETERS:  Count               - Count of objects in the list
 *              Objects             - Array of ACPI_OBJECTs to be deleted
 *
 * RETURN:      None
 *
 * DESCRIPTION: Delete a list of ACPI_OBJECTS. Handles packages and nested
 *              packages via recursion.
 *
 *****************************************************************************/

void
AcpiDbDeleteObjects (
    UINT32                  Count,
    ACPI_OBJECT             *Objects)
{
    UINT32                  i;


    for (i = 0; i < Count; i++)
    {
        switch (Objects[i].Type)
        {
        case ACPI_TYPE_BUFFER:

            ACPI_FREE (Objects[i].Buffer.Pointer);
            break;

        case ACPI_TYPE_PACKAGE:

            /* Recursive call to delete package elements */

            AcpiDbDeleteObjects (Objects[i].Package.Count,
                Objects[i].Package.Elements);

            /* Free the elements array */

            ACPI_FREE (Objects[i].Package.Elements);
```

```
126             break;

128         default:

130             break;
131         }
132     }
133 }


136 /******************************************************************************
137  *
138  * FUNCTION:    AcpiDbExecuteMethod
139  *
140  * PARAMETERS:  Info            - Valid info segment
141  *              ReturnObj       - Where to put return object
142  *
143  * RETURN:      Status
144  *
145  * DESCRIPTION: Execute a control method.
146  *
147  *****************************************************************************/

149 static ACPI_STATUS
150 AcpiDbExecuteMethod (
151     ACPI_DB_METHOD_INFO     *Info,
152     ACPI_BUFFER             *ReturnObj)
153 {
154     ACPI_STATUS             Status;
155     ACPI_OBJECT_LIST        ParamObjects;
156     ACPI_OBJECT             Params[ACPI_DEBUGGER_MAX_ARGS + 1];
157     UINT32                  i;


160     ACPI_FUNCTION_TRACE (DbExecuteMethod);


163     if (AcpiGbl_DbOutputToFile && !AcpiDbgLevel)
164     {
165         AcpiOsPrintf ("Warning: debug output is not enabled!\n");
166     }

168     ParamObjects.Count = 0;
169     ParamObjects.Pointer = NULL;

171     /* Pass through any command-line arguments */

173     if (Info->Args && Info->Args[0])
174     {
175         /* Get arguments passed on the command line */

177         for (i = 0; (Info->Args[i] && *(Info->Args[i])); i++)
178         {
179             /* Convert input string (token) to an actual ACPI_OBJECT */

181             Status = AcpiDbConvertToObject (Info->Types[i],
182                 Info->Args[i], &Params[i]);
183             if (ACPI_FAILURE (Status))
184             {
185                 ACPI_EXCEPTION ((AE_INFO, Status,
186                     "While parsing method arguments"));
187                 goto Cleanup;
188             }
189         }

191         ParamObjects.Count = i;
```



```
192         ParamObjects.Pointer = Params;
193     }

195     /* Prepare for a return object of arbitrary size */

197     ReturnObj->Pointer = AcpiGbl_DbBuffer;
198     ReturnObj->Length  = ACPI_DEBUG_BUFFER_SIZE;

200     /* Do the actual method execution */

202     AcpiGbl_MethodExecuting = TRUE;
203     Status = AcpiEvaluateObject (NULL, Info->Pathname,
204         &ParamObjects, ReturnObj);

206     AcpiGbl_CmSingleStep = FALSE;
207     AcpiGbl_MethodExecuting = FALSE;

209     if (ACPI_FAILURE (Status))
210     {
211         ACPI_EXCEPTION ((AE_INFO, Status,
212             "while executing %s from debugger", Info->Pathname));

214         if (Status == AE_BUFFER_OVERFLOW)
215         {
216             ACPI_ERROR ((AE_INFO,
217                 "Possible overflow of internal debugger buffer (size 0x%X needed
218                 ACPI_DEBUG_BUFFER_SIZE, (UINT32) ReturnObj->Length));
219         }
220     }

222 Cleanup:
223     AcpiDbDeleteObjects (ParamObjects.Count, Params);
224     return_ACPI_STATUS (Status);
225 }


228 /******************************************************************************
229  *
230  * FUNCTION:    AcpiDbExecuteSetup
231  *
232  * PARAMETERS:  Info            - Valid method info
233  *
234  * RETURN:      None
235  *
236  * DESCRIPTION: Setup info segment prior to method execution
237  *
238  *****************************************************************************/

240 static ACPI_STATUS
241 AcpiDbExecuteSetup (
242     ACPI_DB_METHOD_INFO     *Info)
243 {
244     ACPI_STATUS             Status;


247     ACPI_FUNCTION_NAME (DbExecuteSetup);


250     /* Catenate the current scope to the supplied name */

252     Info->Pathname[0] = 0;
253     if ((Info->Name[0] != '\\') &&
254         (Info->Name[0] != '/'))
255     {
256         if (AcpiUtSafeStrcat (Info->Pathname, sizeof (Info->Pathname),
257             AcpiGbl_DbScopeBuf))
```

```
258         {
259             Status = AE_BUFFER_OVERFLOW;
260             goto ErrorExit;
261         }
262     }

264     if (AcpiUtSafeStrcat (Info->Pathname, sizeof (Info->Pathname),
265         Info->Name))
266     {
267         Status = AE_BUFFER_OVERFLOW;
268         goto ErrorExit;
269     }

271     AcpiDbPrepNamestring (Info->Pathname);

273     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
274     AcpiOsPrintf ("Evaluating %s\n", Info->Pathname);

276     if (Info->Flags & EX_SINGLE_STEP)
277     {
278         AcpiGbl_CmSingleStep = TRUE;
279         AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
280     }

282     else
283     {
284         /* No single step, allow redirection to a file */

286         AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
287     }

289     return (AE_OK);

291 ErrorExit:

293     ACPI_EXCEPTION ((AE_INFO, Status, "During setup for method execution"));
294     return (Status);
295 }


298 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
299 UINT32
300 AcpiDbGetCacheInfo (
301     ACPI_MEMORY_LIST        *Cache)
302 {

304     return (Cache->TotalAllocated - Cache->TotalFreed - Cache->CurrentDepth);
305 }
306 #endif

308 /*******************************************************************************
309  *
310  * FUNCTION:    AcpiDbGetOutstandingAllocations
311  *
312  * PARAMETERS:  None
313  *
314  * RETURN:      Current global allocation count minus cache entries
315  *
316  * DESCRIPTION: Determine the current number of "outstanding" allocations --
317  *              those allocations that have not been freed and also are not
318  *              in one of the various object caches.
319  *
320  ******************************************************************************/

322 static UINT32
323 AcpiDbGetOutstandingAllocations (
```



```
324     void)
325 {
326     UINT32                  Outstanding = 0;

328 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

330     Outstanding += AcpiDbGetCacheInfo (AcpiGbl_StateCache);
331     Outstanding += AcpiDbGetCacheInfo (AcpiGbl_PsNodeCache);
332     Outstanding += AcpiDbGetCacheInfo (AcpiGbl_PsNodeExtCache);
333     Outstanding += AcpiDbGetCacheInfo (AcpiGbl_OperandCache);
334 #endif

336     return (Outstanding);
337 }


340 /*******************************************************************************
341  *
342  * FUNCTION:    AcpiDbExecutionWalk
343  *
344  * PARAMETERS:  WALK_CALLBACK
345  *
346  * RETURN:      Status
347  *
348  * DESCRIPTION: Execute a control method. Name is relative to the current
349  *              scope.
350  *
351  ******************************************************************************/

353 static ACPI_STATUS
354 AcpiDbExecutionWalk (
355     ACPI_HANDLE             ObjHandle,
356     UINT32                  NestingLevel,
357     void                    *Context,
358     void                    **ReturnValue)
359 {
360     ACPI_OPERAND_OBJECT     *ObjDesc;
361     ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
362     ACPI_BUFFER             ReturnObj;
363     ACPI_STATUS             Status;


366     ObjDesc = AcpiNsGetAttachedObject (Node);
367     if (ObjDesc->Method.ParamCount)
368     {
369         return (AE_OK);
370     }

372     ReturnObj.Pointer = NULL;
373     ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

375     AcpiNsPrintNodePathname (Node, "Evaluating");

377     /* Do the actual method execution */

379     AcpiOsPrintf ("\n");
380     AcpiGbl_MethodExecuting = TRUE;

382     Status = AcpiEvaluateObject (Node, NULL, NULL, &ReturnObj);

384     AcpiOsPrintf ("Evaluation of [%4.4s] returned %s\n", AcpiUtGetNodeName (Node
385             AcpiFormatException (Status));
386     AcpiGbl_MethodExecuting = FALSE;

388     return (AE_OK);
389 }
```

```
392 /*****************************************************************************
393  *
394  * FUNCTION:    AcpiDbExecute
395  *
396  * PARAMETERS:  Name                - Name of method to execute
397  *              Args                - Parameters to the method
398  *              Flags               - single step/no single step
399  *
400  * RETURN:      None
401  *
402  * DESCRIPTION: Execute a control method. Name is relative to the current
403  *              scope.
404  *
405  ****************************************************************************/

407 void
408 AcpiDbExecute (
409     char                    *Name,
410     char                    **Args,
411     ACPI_OBJECT_TYPE        *Types,
412     UINT32                  Flags)
413 {
414     ACPI_STATUS             Status;
415     ACPI_BUFFER             ReturnObj;
416     char                    *NameString;


419 #ifdef ACPI_DEBUG_OUTPUT
420     UINT32                  PreviousAllocations;
421     UINT32                  Allocations;


424     /* Memory allocation tracking */

426     PreviousAllocations = AcpiDbGetOutstandingAllocations ();
427 #endif

429     if (*Name == '*')
430     {
431         (void) AcpiWalkNamespace (ACPI_TYPE_METHOD, ACPI_ROOT_OBJECT,
432                     ACPI_UINT32_MAX, AcpiDbExecutionWalk, NULL, NULL, NULL);
433         return;
434     }
435     else
436     {
437         NameString = ACPI_ALLOCATE (ACPI_STRLEN (Name) + 1);
438         if (!NameString)
439         {
440             return;
441         }

443         ACPI_MEMSET (&AcpiGbl_DbMethodInfo, 0, sizeof (ACPI_DB_METHOD_INFO));

445         ACPI_STRCPY (NameString, Name);
446         AcpiUtStrupr (NameString);
447         AcpiGbl_DbMethodInfo.Name = NameString;
448         AcpiGbl_DbMethodInfo.Args = Args;
449         AcpiGbl_DbMethodInfo.Types = Types;
450         AcpiGbl_DbMethodInfo.Flags = Flags;

452         ReturnObj.Pointer = NULL;
453         ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

455         Status = AcpiDbExecuteSetup (&AcpiGbl_DbMethodInfo);
```



```
456         if (ACPI_FAILURE (Status))
457         {
458             ACPI_FREE (NameString);
459             return;
460         }

462         /* Get the NS node, determines existence also */

464         Status = AcpiGetHandle (NULL, AcpiGbl_DbMethodInfo.Pathname,
465             &AcpiGbl_DbMethodInfo.Method);
466         if (ACPI_SUCCESS (Status))
467         {
468             Status = AcpiDbExecuteMethod (&AcpiGbl_DbMethodInfo, &ReturnObj);
469         }
470         ACPI_FREE (NameString);
471     }

473     /*
474      * Allow any handlers in separate threads to complete.
475      * (Such as Notify handlers invoked from AML executed above).
476      */
477     AcpiOsSleep ((UINT64) 10);

479 #ifdef ACPI_DEBUG_OUTPUT

481     /* Memory allocation tracking */

483     Allocations = AcpiDbGetOutstandingAllocations () - PreviousAllocations;

485     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);

487     if (Allocations > 0)
488     {
489         AcpiOsPrintf ("0x%X Outstanding allocations after evaluation of %s\n",
490                       Allocations, AcpiGbl_DbMethodInfo.Pathname);
491     }
492 #endif

494     if (ACPI_FAILURE (Status))
495     {
496         AcpiOsPrintf ("Evaluation of %s failed with status %s\n",
497             AcpiGbl_DbMethodInfo.Pathname, AcpiFormatException (Status));
498     }
499     else
500     {
501         /* Display a return object, if any */

503         if (ReturnObj.Length)
504         {
505             AcpiOsPrintf (
506                 "Evaluation of %s returned object %p, external buffer length %X\
507                 AcpiGbl_DbMethodInfo.Pathname, ReturnObj.Pointer,
508                 (UINT32) ReturnObj.Length);
509             AcpiDbDumpExternalObject (ReturnObj.Pointer, 1);

511             /* Dump a _PLD buffer if present */

513             if (ACPI_COMPARE_NAME ((ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
514                     AcpiGbl_DbMethodInfo.Method)->Name.Ascii), METHOD_NAME__PLD)
515             {
516                 AcpiDbDumpPldBuffer (ReturnObj.Pointer);
517             }
518         }
519         else
520         {
521             AcpiOsPrintf ("No object was returned from evaluation of %s\n",
```

```
522                 AcpiGbl_DbMethodInfo.Pathname);
523         }
524     }

526     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
527 }


530 /******************************************************************************
531  *
532  * FUNCTION:    AcpiDbMethodThread
533  *
534  * PARAMETERS:  Context             - Execution info segment
535  *
536  * RETURN:      None
537  *
538  * DESCRIPTION: Debugger execute thread. Waits for a command line, then
539  *              simply dispatches it.
540  *
541  *****************************************************************************/

543 static void ACPI_SYSTEM_XFACE
544 AcpiDbMethodThread (
545     void                    *Context)
546 {
547     ACPI_STATUS             Status;
548     ACPI_DB_METHOD_INFO     *Info = Context;
549     ACPI_DB_METHOD_INFO     LocalInfo;
550     UINT32                  i;
551     UINT8                   Allow;
552     ACPI_BUFFER             ReturnObj;


555     /*
556      * AcpiGbl_DbMethodInfo.Arguments will be passed as method arguments.
557      * Prevent AcpiGbl_DbMethodInfo from being modified by multiple threads
558      * concurrently.
559      *
560      * Note: The arguments we are passing are used by the ASL test suite
561      * (aslts). Do not change them without updating the tests.
562      */
563     (void) AcpiOsWaitSemaphore (Info->InfoGate, 1, ACPI_WAIT_FOREVER);

565     if (Info->InitArgs)
566     {
567         AcpiDbUint32ToHexString (Info->NumCreated, Info->IndexOfThreadStr);
568         AcpiDbUint32ToHexString ((UINT32) AcpiOsGetThreadId (), Info->IdOfThread
569     }

571     if (Info->Threads && (Info->NumCreated < Info->NumThreads))
572     {
573         Info->Threads[Info->NumCreated++] = AcpiOsGetThreadId();
574     }

576     LocalInfo = *Info;
577     LocalInfo.Args = LocalInfo.Arguments;
578     LocalInfo.Arguments[0] = LocalInfo.NumThreadsStr;
579     LocalInfo.Arguments[1] = LocalInfo.IdOfThreadStr;
580     LocalInfo.Arguments[2] = LocalInfo.IndexOfThreadStr;
581     LocalInfo.Arguments[3] = NULL;

583     LocalInfo.Types = LocalInfo.ArgTypes;

585     (void) AcpiOsSignalSemaphore (Info->InfoGate, 1);

587     for (i = 0; i < Info->NumLoops; i++)
```



```
588     {
589         Status = AcpiDbExecuteMethod (&LocalInfo, &ReturnObj);
590         if (ACPI_FAILURE (Status))
591         {
592             AcpiOsPrintf ("%s During evaluation of %s at iteration %X\n",
593                 AcpiFormatException (Status), Info->Pathname, i);
594             if (Status == AE_ABORT_METHOD)
595             {
596                 break;
597             }
598         }

600 #if 0
601         if ((i % 100) == 0)
602         {
603             AcpiOsPrintf ("%u loops, Thread 0x%x\n", i, AcpiOsGetThreadId ());
604         }

606         if (ReturnObj.Length)
607         {
608             AcpiOsPrintf ("Evaluation of %s returned object %p Buflen %X\n",
609                 Info->Pathname, ReturnObj.Pointer, (UINT32) ReturnObj.Length);
610             AcpiDbDumpExternalObject (ReturnObj.Pointer, 1);
611         }
612 #endif
613     }

615     /* Signal our completion */

617     Allow = 0;
618     (void) AcpiOsWaitSemaphore (Info->ThreadCompleteGate, 1, ACPI_WAIT_FOREVER);
619     Info->NumCompleted++;

621     if (Info->NumCompleted == Info->NumThreads)
622     {
623         /* Do signal for main thread once only */
624         Allow = 1;
625     }

627     (void) AcpiOsSignalSemaphore (Info->ThreadCompleteGate, 1);

629     if (Allow)
630     {
631         Status = AcpiOsSignalSemaphore (Info->MainThreadGate, 1);
632         if (ACPI_FAILURE (Status))
633         {
634             AcpiOsPrintf ("Could not signal debugger thread sync semaphore, %s\n
635                 AcpiFormatException (Status));
636         }
637     }
638 }


641 /******************************************************************************
642  *
643  * FUNCTION:    AcpiDbCreateExecutionThreads
644  *
645  * PARAMETERS:  NumThreadsArg           - Number of threads to create
646  *              NumLoopsArg             - Loop count for the thread(s)
647  *              MethodNameArg           - Control method to execute
648  *
649  * RETURN:      None
650  *
651  * DESCRIPTION: Create threads to execute method(s)
652  *
653  *****************************************************************************/
```

```
655 void
656 AcpiDbCreateExecutionThreads (
657     char                    *NumThreadsArg,
658     char                    *NumLoopsArg,
659     char                    *MethodNameArg)
660 {
661     ACPI_STATUS             Status;
662     UINT32                  NumThreads;
663     UINT32                  NumLoops;
664     UINT32                  i;
665     UINT32                  Size;
666     ACPI_MUTEX              MainThreadGate;
667     ACPI_MUTEX              ThreadCompleteGate;
668     ACPI_MUTEX              InfoGate;


671     /* Get the arguments */

673     NumThreads = ACPI_STRTOUL (NumThreadsArg, NULL, 0);
674     NumLoops   = ACPI_STRTOUL (NumLoopsArg, NULL, 0);

676     if (!NumThreads || !NumLoops)
677     {
678         AcpiOsPrintf ("Bad argument: Threads %X, Loops %X\n",
679             NumThreads, NumLoops);
680         return;
681     }

683     /*
684      * Create the semaphore for synchronization of
685      * the created threads with the main thread.
686      */
687     Status = AcpiOsCreateSemaphore (1, 0, &MainThreadGate);
688     if (ACPI_FAILURE (Status))
689     {
690         AcpiOsPrintf ("Could not create semaphore for synchronization with the m
691             AcpiFormatException (Status));
692         return;
693     }

695     /*
696      * Create the semaphore for synchronization
697      * between the created threads.
698      */
699     Status = AcpiOsCreateSemaphore (1, 1, &ThreadCompleteGate);
700     if (ACPI_FAILURE (Status))
701     {
702         AcpiOsPrintf ("Could not create semaphore for synchronization between th
703             AcpiFormatException (Status));
704         (void) AcpiOsDeleteSemaphore (MainThreadGate);
705         return;
706     }

708     Status = AcpiOsCreateSemaphore (1, 1, &InfoGate);
709     if (ACPI_FAILURE (Status))
710     {
711         AcpiOsPrintf ("Could not create semaphore for synchronization of AcpiGbl
712             AcpiFormatException (Status));
713         (void) AcpiOsDeleteSemaphore (ThreadCompleteGate);
714         (void) AcpiOsDeleteSemaphore (MainThreadGate);
715         return;
716     }

718     ACPI_MEMSET (&AcpiGbl_DbMethodInfo, 0, sizeof (ACPI_DB_METHOD_INFO));
```



```
720     /* Array to store IDs of threads */

722     AcpiGbl_DbMethodInfo.NumThreads = NumThreads;
723     Size = sizeof (ACPI_THREAD_ID) * AcpiGbl_DbMethodInfo.NumThreads;
724     AcpiGbl_DbMethodInfo.Threads = AcpiOsAllocate (Size);
725     if (AcpiGbl_DbMethodInfo.Threads == NULL)
726     {
727         AcpiOsPrintf ("No memory for thread IDs array\n");
728         (void) AcpiOsDeleteSemaphore (MainThreadGate);
729         (void) AcpiOsDeleteSemaphore (ThreadCompleteGate);
730         (void) AcpiOsDeleteSemaphore (InfoGate);
731         return;
732     }
733     ACPI_MEMSET (AcpiGbl_DbMethodInfo.Threads, 0, Size);

735     /* Setup the context to be passed to each thread */

737     AcpiGbl_DbMethodInfo.Name = MethodNameArg;
738     AcpiGbl_DbMethodInfo.Flags = 0;
739     AcpiGbl_DbMethodInfo.NumLoops = NumLoops;
740     AcpiGbl_DbMethodInfo.MainThreadGate = MainThreadGate;
741     AcpiGbl_DbMethodInfo.ThreadCompleteGate = ThreadCompleteGate;
742     AcpiGbl_DbMethodInfo.InfoGate = InfoGate;

744     /* Init arguments to be passed to method */

746     AcpiGbl_DbMethodInfo.InitArgs = 1;
747     AcpiGbl_DbMethodInfo.Args = AcpiGbl_DbMethodInfo.Arguments;
748     AcpiGbl_DbMethodInfo.Arguments[0] = AcpiGbl_DbMethodInfo.NumThreadsStr;
749     AcpiGbl_DbMethodInfo.Arguments[1] = AcpiGbl_DbMethodInfo.IdOfThreadStr;
750     AcpiGbl_DbMethodInfo.Arguments[2] = AcpiGbl_DbMethodInfo.IndexOfThreadStr;
751     AcpiGbl_DbMethodInfo.Arguments[3] = NULL;

753     AcpiGbl_DbMethodInfo.Types = AcpiGbl_DbMethodInfo.ArgTypes;
754     AcpiGbl_DbMethodInfo.ArgTypes[0] = ACPI_TYPE_INTEGER;
755     AcpiGbl_DbMethodInfo.ArgTypes[1] = ACPI_TYPE_INTEGER;
756     AcpiGbl_DbMethodInfo.ArgTypes[2] = ACPI_TYPE_INTEGER;

758     AcpiDbUint32ToHexString (NumThreads, AcpiGbl_DbMethodInfo.NumThreadsStr);

760     Status = AcpiDbExecuteSetup (&AcpiGbl_DbMethodInfo);
761     if (ACPI_FAILURE (Status))
762     {
763         goto CleanupAndExit;
764     }

766     /* Get the NS node, determines existence also */

768     Status = AcpiGetHandle (NULL, AcpiGbl_DbMethodInfo.Pathname,
769         &AcpiGbl_DbMethodInfo.Method);
770     if (ACPI_FAILURE (Status))
771     {
772         AcpiOsPrintf ("%s Could not get handle for %s\n",
773             AcpiFormatException (Status), AcpiGbl_DbMethodInfo.Pathname);
774         goto CleanupAndExit;
775     }

777     /* Create the threads */

779     AcpiOsPrintf ("Creating %X threads to execute %X times each\n",
780         NumThreads, NumLoops);

782     for (i = 0; i < (NumThreads); i++)
783     {
784         Status = AcpiOsExecute (OSL_DEBUGGER_THREAD, AcpiDbMethodThread,
785             &AcpiGbl_DbMethodInfo);
```

```
786         if (ACPI_FAILURE (Status))
787         {
788             break;
789         }
790     }

792     /* Wait for all threads to complete */

794     (void) AcpiOsWaitSemaphore (MainThreadGate, 1, ACPI_WAIT_FOREVER);

796     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
797     AcpiOsPrintf ("All threads (%X) have completed\n", NumThreads);
798     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);

800 CleanupAndExit:

802     /* Cleanup and exit */

804     (void) AcpiOsDeleteSemaphore (MainThreadGate);
805     (void) AcpiOsDeleteSemaphore (ThreadCompleteGate);
806     (void) AcpiOsDeleteSemaphore (InfoGate);

808     AcpiOsFree (AcpiGbl_DbMethodInfo.Threads);
809     AcpiGbl_DbMethodInfo.Threads = NULL;
810 }

812 #endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   15886 Thu Dec 26 13:48:42 2013
new/usr/src/common/acpica/components/debugger/dbfileio.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dbfileio - Debugger file I/O commands. These can't usually
   4  *              be used when running the debugger in Ring 0 (Kernel mode)
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */


  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acdebug.h"

  50 #ifdef ACPI_APPLICATION
  51 #include "actables.h"
  52 #endif

  54 #ifdef ACPI_ASL_COMPILER
  55 #include "aslcompiler.h"
  56 #endif
```



```
  58 #if (defined ACPI_DEBUGGER || defined ACPI_DISASSEMBLER)

  60 #define _COMPONENT          ACPI_CA_DEBUGGER
  61         ACPI_MODULE_NAME    ("dbfileio")

  59 /*
  60  * NOTE: this is here for lack of a better place. It is used in all
  61  * flavors of the debugger, need LCD file
  62  */
  63 #ifdef ACPI_APPLICATION
  64 #include <stdio.h>
  65 FILE                        *AcpiGbl_DebugFile = NULL;
  66 #endif


  63 #ifdef ACPI_DEBUGGER

  65 /* Local prototypes */

  67 #ifdef ACPI_APPLICATION

  69 static ACPI_STATUS
  70 AcpiDbCheckTextModeCorruption (
  71     UINT8                   *Table,
  72     UINT32                  TableLength,
  73     UINT32                  FileLength);

  75 #endif

  77 /******************************************************************************
  78  *
  79  * FUNCTION:    AcpiDbCloseDebugFile
  80  *
  81  * PARAMETERS:  None
  82  *
  83  * RETURN:      None
  84  *
  85  * DESCRIPTION: If open, close the current debug output file
  86  *
  87  *****************************************************************************/

  89 void
  90 AcpiDbCloseDebugFile (
  91     void)
  92 {

  94 #ifdef ACPI_APPLICATION

  96     if (AcpiGbl_DebugFile)
  97     {
  98        fclose (AcpiGbl_DebugFile);
  99        AcpiGbl_DebugFile = NULL;
 100        AcpiGbl_DbOutputToFile = FALSE;
 101        AcpiOsPrintf ("Debug output file %s closed\n", AcpiGbl_DbDebugFilename);
 102     }
 103 #endif
 104 }


 107 /******************************************************************************
 108  *
 109  * FUNCTION:    AcpiDbOpenDebugFile
 110  *
 111  * PARAMETERS:  Name                - Filename to open
 112  *
 113  * RETURN:      None
```

```
 114  *
 115  * DESCRIPTION: Open a file where debug output will be directed.
 116  *
 117  ******************************************************************************/

 119 void
 120 AcpiDbOpenDebugFile (
 121     char                    *Name)
 122 {

 124 #ifdef ACPI_APPLICATION

 126     AcpiDbCloseDebugFile ();
 127     AcpiGbl_DebugFile = fopen (Name, "w+");
 128     if (!AcpiGbl_DebugFile)
 134     if (AcpiGbl_DebugFile)
 129     {
 136         AcpiOsPrintf ("Debug output file %s opened\n", Name);
 137         ACPI_STRCPY (AcpiGbl_DbDebugFilename, Name);
 138         AcpiGbl_DbOutputToFile = TRUE;
 139     }
 140     else
 141     {
 130         AcpiOsPrintf ("Could not open debug file %s\n", Name);
 131         return;
 132     }

 134     AcpiOsPrintf ("Debug output file %s opened\n", Name);
 135     ACPI_STRNCPY (AcpiGbl_DbDebugFilename, Name,
 136         sizeof (AcpiGbl_DbDebugFilename));
 137     AcpiGbl_DbOutputToFile = TRUE;

 139 #endif
 140 }
_____unchanged_portion_omitted_


 220 /******************************************************************************
 221  *
 222  * FUNCTION:    AcpiDbReadTable
 223  *
 224  * PARAMETERS:  fp              - File that contains table
 225  *              Table           - Return value, buffer with table
 226  *              TableLength     - Return value, length of table
 227  *
 228  * RETURN:      Status
 229  *
 230  * DESCRIPTION: Load the DSDT from the file pointer
 231  *
 232  *****************************************************************************/

 234 static ACPI_STATUS
 235 AcpiDbReadTable (
 236     FILE                    *fp,
 237     ACPI_TABLE_HEADER       **Table,
 238     UINT32                  *TableLength)
 239 {
 240     ACPI_TABLE_HEADER       TableHeader;
 241     UINT32                  Actual;
 242     ACPI_STATUS             Status;
 243     UINT32                  FileSize;
 244     BOOLEAN                 StandardHeader = TRUE;


 247     /* Get the file size */
```



```
 249     fseek (fp, 0, SEEK_END);
 250     FileSize = (UINT32) ftell (fp);
 251     fseek (fp, 0, SEEK_SET);

 253     if (FileSize < 4)
 254     {
 255         return (AE_BAD_HEADER);
 256     }

 258     /* Read the signature */

 260     if (fread (&TableHeader, 1, 4, fp) != 4)
 261     {
 262         AcpiOsPrintf ("Could not read the table signature\n");
 263         return (AE_BAD_HEADER);
 264     }

 266     fseek (fp, 0, SEEK_SET);

 268     /* The RSDP table does not have standard ACPI header */
 274     /* The RSDT and FACS tables do not have standard ACPI headers */

 270     if (ACPI_COMPARE_NAME (TableHeader.Signature, "RSD "))
 276     if (ACPI_COMPARE_NAME (TableHeader.Signature, "RSD ") ||
 277         ACPI_COMPARE_NAME (TableHeader.Signature, "FACS"))
 271     {
 272         *TableLength = FileSize;
 273         StandardHeader = FALSE;
 274     }
 275     else
 276     {
 277         /* Read the table header */

 279         if (fread (&TableHeader, 1, sizeof (ACPI_TABLE_HEADER), fp) !=
 286         if (fread (&TableHeader, 1, sizeof (TableHeader), fp) !=
 280                 sizeof (ACPI_TABLE_HEADER))
 281         {
 282             AcpiOsPrintf ("Could not read the table header\n");
 283             return (AE_BAD_HEADER);
 284         }

 286 #if 0
 287         /* Validate the table header/length */

 289         Status = AcpiTbValidateTableHeader (&TableHeader);
 290         if (ACPI_FAILURE (Status))
 291         {
 292             AcpiOsPrintf ("Table header is invalid!\n");
 293             return (Status);
 294         }
 295 #endif

 297         /* File size must be at least as long as the Header-specified length */

 299         if (TableHeader.Length > FileSize)
 300         {
 301             AcpiOsPrintf (
 302                 "TableHeader length [0x%X] greater than the input file size [0x%
 303                 TableHeader.Length, FileSize);

 305 #ifdef ACPI_ASL_COMPILER
 306             Status = FlCheckForAscii (fp, NULL, FALSE);
 307             if (ACPI_SUCCESS (Status))
 308             {
 309                 AcpiOsPrintf ("File appears to be ASCII only, must be binary\n",
 310                     TableHeader.Length, FileSize);
```

```
311             }
312 #endif
313             return (AE_BAD_HEADER);
314         }

316 #ifdef ACPI_OBSOLETE_CODE
317         /* We only support a limited number of table types */

319         if (!ACPI_COMPARE_NAME ((char *) TableHeader.Signature, ACPI_SIG_DSDT) &
320             !ACPI_COMPARE_NAME ((char *) TableHeader.Signature, ACPI_SIG_PSDT) &
321             !ACPI_COMPARE_NAME ((char *) TableHeader.Signature, ACPI_SIG_SSDT))
317         if (ACPI_STRNCMP ((char *) TableHeader.Signature, DSDT_SIG, 4) &&
318             ACPI_STRNCMP ((char *) TableHeader.Signature, PSDT_SIG, 4) &&
319             ACPI_STRNCMP ((char *) TableHeader.Signature, SSDT_SIG, 4))
322         {
323             AcpiOsPrintf ("Table signature [%4.4s] is invalid or not supported\n
324                 (char *) TableHeader.Signature);
325             ACPI_DUMP_BUFFER (&TableHeader, sizeof (ACPI_TABLE_HEADER));
326             return (AE_ERROR);
327         }
328 #endif

330         *TableLength = TableHeader.Length;
331     }

333     /* Allocate a buffer for the table */

335     *Table = AcpiOsAllocate ((size_t) FileSize);
336     if (!*Table)
337     {
338         AcpiOsPrintf (
339             "Could not allocate memory for ACPI table %4.4s (size=0x%X)\n",
340             TableHeader.Signature, *TableLength);
341         return (AE_NO_MEMORY);
342     }

344     /* Get the rest of the table */

346     fseek (fp, 0, SEEK_SET);
347     Actual = fread (*Table, 1, (size_t) FileSize, fp);
348     if (Actual == FileSize)
349     {
350         if (StandardHeader)
351         {
352             /* Now validate the checksum */

354             Status = AcpiTbVerifyChecksum ((void *) *Table,
355                         ACPI_CAST_PTR (ACPI_TABLE_HEADER, *Table)->Length);

357             if (Status == AE_BAD_CHECKSUM)
358             {
359                 Status = AcpiDbCheckTextModeCorruption ((UINT8 *) *Table,
360                             FileSize, (*Table)->Length);
361                 return (Status);
362             }
363         }
364         return (AE_OK);
365     }

367     if (Actual > 0)
368     {
369         AcpiOsPrintf ("Warning - reading table, asked for %X got %X\n",
370             FileSize, Actual);
371         return (AE_OK);
372     }
```



```
374     AcpiOsPrintf ("Error - could not read the table file\n");
375     AcpiOsFree (*Table);
376     *Table = NULL;
377     *TableLength = 0;

378     return (AE_ERROR);
379 }
______unchanged_portion_omitted_


457 /******************************************************************************
458  *
459  * FUNCTION:    AcpiDbReadTableFromFile
460  *
461  * PARAMETERS:  Filename         - File where table is located
462  *              Table            - Where a pointer to the table is returned
463  *
464  * RETURN:      Status
465  *
466  * DESCRIPTION: Get an ACPI table from a file
467  *
468  *****************************************************************************/

470 ACPI_STATUS
471 AcpiDbReadTableFromFile (
472     char                    *Filename,
473     ACPI_TABLE_HEADER       **Table)
474 {
475     FILE                    *File;
474     FILE                    *fp;
476     UINT32                  TableLength;
477     ACPI_STATUS             Status;


480     /* Open the file */

482     File = fopen (Filename, "rb");
483     if (!File)
481     fp = fopen (Filename, "rb");
482     if (!fp)
484     {
485         perror ("Could not open input file");
484         AcpiOsPrintf ("Could not open input file %s\n", Filename);
486         return (AE_ERROR);
487     }

489     /* Get the entire file */

491     fprintf (stderr, "Loading Acpi table from file %s\n", Filename);
492     Status = AcpiDbReadTable (File, Table, &TableLength);
493     fclose(File);
491     Status = AcpiDbReadTable (fp, Table, &TableLength);
492     fclose(fp);

495     if (ACPI_FAILURE (Status))
496     {
497         AcpiOsPrintf ("Could not get table from the file\n");
498         return (Status);
499     }

501     return (AE_OK);
502 }
503 #endif


506 /******************************************************************************
```

```
 *
 * FUNCTION:    AcpiDbGetTableFromFile
 *
 * PARAMETERS:  Filename        - File where table is located
 *              ReturnTable     - Where a pointer to the table is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Load an ACPI table from a file
 *
 ******************************************************************************/

ACPI_STATUS
AcpiDbGetTableFromFile (
    char                    *Filename,
    ACPI_TABLE_HEADER       **ReturnTable)
{
#ifdef ACPI_APPLICATION
    ACPI_STATUS             Status;
    ACPI_TABLE_HEADER       *Table;
    BOOLEAN                 IsAmlTable = TRUE;


    Status = AcpiDbReadTableFromFile (Filename, &Table);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

#ifdef ACPI_DATA_TABLE_DISASSEMBLY
    IsAmlTable = AcpiUtIsAmlTable (Table);
#endif

    if (IsAmlTable)
    {
        /* Attempt to recognize and install the table */

        Status = AeLocalLoadTable (Table);
        if (ACPI_FAILURE (Status))
        {
            if (Status == AE_ALREADY_EXISTS)
            {
                AcpiOsPrintf ("Table %4.4s is already installed\n",
                    Table->Signature);
            }
            else
            {
                AcpiOsPrintf ("Could not install table, %s\n",
                    AcpiFormatException (Status));
            }

            return (Status);
        }

        fprintf (stderr,
            "Acpi table [%4.4s] successfully installed and loaded\n",
            Table->Signature);
    }

    AcpiGbl_AcpiHardwarePresent = FALSE;
    if (ReturnTable)
    {
        *ReturnTable = Table;
    }
```



```
#endif  /* ACPI_APPLICATION */
    return (AE_OK);
}

#endif  /* ACPI_DEBUGGER */
```

```
**********************************************************
    7672 Thu Dec 26 13:48:42 2013
new/usr/src/common/acpica/components/debugger/dbhistry.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbhistry - debugger HISTORY command
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdebug.h"

#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbhistry")


#define HI_NO_HISTORY       0
#define HI_RECORD_HISTORY   1
#define HISTORY_SIZE        40
```



```
typedef struct HistoryInfo
{
    char                    *Command;
    UINT32                  CmdNum;

} HISTORY_INFO;


static HISTORY_INFO         AcpiGbl_HistoryBuffer[HISTORY_SIZE];
static UINT16               AcpiGbl_LoHistory = 0;
static UINT16               AcpiGbl_NumHistory = 0;
static UINT16               AcpiGbl_NextHistoryIndex = 0;
UINT32                      AcpiGbl_NextCmdNum = 1;


/******************************************************************************
 *
 * FUNCTION:    AcpiDbAddToHistory
 *
 * PARAMETERS:  CommandLine     - Command to add
 *
 * RETURN:      None
 *
 * DESCRIPTION: Add a command line to the history buffer.
 *
 *****************************************************************************/

void
AcpiDbAddToHistory (
    char                    *CommandLine)
{
    UINT16                  CmdLen;
    UINT16                  BufferLen;

    /* Put command into the next available slot */

    CmdLen = (UINT16) ACPI_STRLEN (CommandLine);
    if (!CmdLen)
    {
        return;
    }

    if (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command != NULL)
    {
        BufferLen = (UINT16) ACPI_STRLEN (
            AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command);
        if (CmdLen > BufferLen)
        {
            AcpiOsFree (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].
                Command);
            AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command =
                AcpiOsAllocate (CmdLen + 1);
        }
    }
    else
    {
        AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command =
            AcpiOsAllocate (CmdLen + 1);
    }

    ACPI_STRCPY (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command,
        CommandLine);

    AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].CmdNum =
        AcpiGbl_NextCmdNum;
```

```
126     /* Adjust indexes */

128     if ((AcpiGbl_NumHistory == HISTORY_SIZE) &&
129         (AcpiGbl_NextHistoryIndex == AcpiGbl_LoHistory))
130     {
131         AcpiGbl_LoHistory++;
132         if (AcpiGbl_LoHistory >= HISTORY_SIZE)
133         {
134             AcpiGbl_LoHistory = 0;
135         }
136     }

138     AcpiGbl_NextHistoryIndex++;
139     if (AcpiGbl_NextHistoryIndex >= HISTORY_SIZE)
140     {
141         AcpiGbl_NextHistoryIndex = 0;
142     }

144     AcpiGbl_NextCmdNum++;
145     if (AcpiGbl_NumHistory < HISTORY_SIZE)
146     {
147         AcpiGbl_NumHistory++;
148     }
149 }


152 /*******************************************************************************
153  *
154  * FUNCTION:    AcpiDbDisplayHistory
155  *
156  * PARAMETERS:  None
157  *
158  * RETURN:      None
159  *
160  * DESCRIPTION: Display the contents of the history buffer
161  *
162  ******************************************************************************/

164 void
165 AcpiDbDisplayHistory (
166     void)
167 {
168     UINT32                  i;
169     UINT16                  HistoryIndex;


172     HistoryIndex = AcpiGbl_LoHistory;

174     /* Dump entire history buffer */

176     for (i = 0; i < AcpiGbl_NumHistory; i++)
177     {
178         if (AcpiGbl_HistoryBuffer[HistoryIndex].Command)
179         {
180             AcpiOsPrintf ("%3ld  %s\n",
181                 AcpiGbl_HistoryBuffer[HistoryIndex].CmdNum,
182                 AcpiGbl_HistoryBuffer[HistoryIndex].Command);
183         }

185         HistoryIndex++;
186         if (HistoryIndex >= HISTORY_SIZE)
187         {
188             HistoryIndex = 0;
189         }
190     }
191 }
```



```
194 /*******************************************************************************
195  *
196  * FUNCTION:    AcpiDbGetFromHistory
197  *
198  * PARAMETERS:  CommandNumArg           - String containing the number of the
199  *                                        command to be retrieved
200  *
201  * RETURN:      Pointer to the retrieved command. Null on error.
202  *
203  * DESCRIPTION: Get a command from the history buffer
204  *
205  ******************************************************************************/

207 char *
208 AcpiDbGetFromHistory (
209     char                    *CommandNumArg)
210 {
211     UINT32                  CmdNum;


214     if (CommandNumArg == NULL)
215     {
216         CmdNum = AcpiGbl_NextCmdNum - 1;
217     }

219     else
220     {
221         CmdNum = ACPI_STRTOUL (CommandNumArg, NULL, 0);
222     }

224     return (AcpiDbGetHistoryByIndex (CmdNum));
225 }


228 /*******************************************************************************
229  *
230  * FUNCTION:    AcpiDbGetHistoryByIndex
231  *
232  * PARAMETERS:  CmdNum              - Index of the desired history entry.
233  *                                    Values are 0...(AcpiGbl_NextCmdNum - 1)
234  *
235  * RETURN:      Pointer to the retrieved command. Null on error.
236  *
237  * DESCRIPTION: Get a command from the history buffer
238  *
239  ******************************************************************************/

241 char *
242 AcpiDbGetHistoryByIndex (
243     UINT32                  CmdNum)
244 {
245     UINT32                  i;
246     UINT16                  HistoryIndex;


249     /* Search history buffer */

251     HistoryIndex = AcpiGbl_LoHistory;
252     for (i = 0; i < AcpiGbl_NumHistory; i++)
253     {
254         if (AcpiGbl_HistoryBuffer[HistoryIndex].CmdNum == CmdNum)
255         {
256             /* Found the command, return it */
```

```
258             return (AcpiGbl_HistoryBuffer[HistoryIndex].Command);
259         }

261         /* History buffer is circular */

263         HistoryIndex++;
264         if (HistoryIndex >= HISTORY_SIZE)
265         {
266             HistoryIndex = 0;
267         }
268     }

270     AcpiOsPrintf ("Invalid history number: %u\n", HistoryIndex);
271     return (NULL);
272 }

274 #endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   34985 Thu Dec 26 13:48:42 2013
new/usr/src/common/acpica/components/debugger/dbinput.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbinput - user front-end to the AML debugger
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdebug.h"


#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbinput")

/* Local prototypes */

static UINT32
AcpiDbGetLine (
```



```
    char                    *InputBuffer);

static UINT32
AcpiDbMatchCommand (
    char                    *UserCommand);

static void
AcpiDbSingleThread (
    void);

static void
AcpiDbDisplayCommandInfo (
    char                    *Command,
    BOOLEAN                 DisplayAll);

static void
AcpiDbDisplayHelp (
    char                    *Command);

static BOOLEAN
AcpiDbMatchCommandHelp (
    char                            *Command,
    const ACPI_DB_COMMAND_HELP      *Help);


/*
 * Top-level debugger commands.
 *
 * This list of commands must match the string table below it
 */
enum AcpiExDebuggerCommands
{
    CMD_NOT_FOUND = 0,
    CMD_NULL,
    CMD_ALLOCATIONS,
    CMD_ARGS,
    CMD_ARGUMENTS,
    CMD_BATCH,
    CMD_BREAKPOINT,
    CMD_BUSINFO,
    CMD_CALL,
    CMD_CLOSE,
    CMD_DEBUG,
    CMD_DISASSEMBLE,
    CMD_DISASM,
    CMD_DUMP,
    CMD_ENABLEACPI,
    CMD_EVALUATE,
    CMD_EVENT,
    CMD_EXECUTE,
    CMD_EXIT,
    CMD_FIND,
    CMD_GO,
    CMD_GPE,
    CMD_GPES,
    CMD_HANDLERS,
    CMD_HELP,
    CMD_HELP2,
    CMD_HISTORY,
    CMD_HISTORY_EXE,
    CMD_HISTORY_LAST,
    CMD_INFORMATION,
    CMD_INTEGRITY,
    CMD_INTO,
    CMD_LEVEL,
    CMD_LIST,
```

```
125     CMD_LOAD,
126     CMD_LOCALS,
127     CMD_LOCKS,
128     CMD_METHODS,
129     CMD_NAMESPACE,
130     CMD_NOTIFY,
131     CMD_OBJECTS,
132     CMD_OPEN,
133     CMD_OSI,
134     CMD_OWNER,
135     CMD_PATHS,
136     CMD_PREDEFINED,
137     CMD_PREFIX,
138     CMD_QUIT,
139     CMD_REFERENCES,
140     CMD_RESOURCES,
141     CMD_RESULTS,
142     CMD_SCI,
143     CMD_SET,
144     CMD_SLEEP,
145     CMD_STATS,
146     CMD_STOP,
147     CMD_TABLES,
148     CMD_TEMPLATE,
149     CMD_TERMINATE,
150     CMD_THREADS,
151     CMD_TRACE,
152     CMD_TREE,
153     CMD_TYPE,
154     CMD_UNLOAD
155 };

157 #define CMD_FIRST_VALID     2


160 /* Second parameter is the required argument count */

162 static const ACPI_DB_COMMAND_INFO   AcpiGbl_DbCommands[] =
163 {
164     {"<NOT FOUND>",  0},
165     {"<NULL>",       0},
166     {"ALLOCATIONS",  0},
167     {"ARGS",         0},
168     {"ARGUMENTS",    0},
169     {"BATCH",        0},
170     {"BREAKPOINT",   1},
171     {"BUSINFO",      0},
172     {"CALL",         0},
173     {"CLOSE",        0},
174     {"DEBUG",        1},
175     {"DISASSEMBLE",  1},
176     {"DISASM",       1},
177     {"DUMP",         1},
178     {"ENABLEACPI",   0},
179     {"EVALUATE",     1},
180     {"EVENT",        1},
181     {"EXECUTE",      1},
182     {"EXIT",         0},
183     {"FIND",         1},
184     {"GO",           0},
185     {"GPE",          2},
186     {"GPES",         0},
187     {"HANDLERS",     0},
188     {"HELP",         0},
189     {"?",            0},
190     {"HISTORY",      0},
```



```
191     {"!",            1},
192     {"!!",           0},
193     {"INFORMATION",  0},
194     {"INTEGRITY",    0},
195     {"INTO",         0},
196     {"LEVEL",        0},
197     {"LIST",         0},
198     {"LOAD",         1},
199     {"LOCALS",       0},
200     {"LOCKS",        0},
201     {"METHODS",      0},
202     {"NAMESPACE",    0},
203     {"NOTIFY",       2},
204     {"OBJECTS",      1},
205     {"OPEN",         1},
206     {"OSI",          0},
207     {"OWNER",        1},
208     {"PATHS",        0},
209     {"PREDEFINED",   0},
210     {"PREFIX",       0},
211     {"QUIT",         0},
212     {"REFERENCES",   1},
213     {"RESOURCES",    0},
214     {"RESULTS",      0},
215     {"SCI",          0},
216     {"SET",          3},
217     {"SLEEP",        0},
218     {"STATS",        1},
219     {"STOP",         0},
220     {"TABLES",       0},
221     {"TEMPLATE",     1},
222     {"TERMINATE",    0},
223     {"THREADS",      3},
224     {"TRACE",        1},
225     {"TREE",         0},
226     {"TYPE",         1},
227     {"UNLOAD",       1},
228     {NULL,           0}
229 };

231 /*
232  * Help for all debugger commands. First argument is the number of lines
233  * of help to output for the command.
234  */
235 static const ACPI_DB_COMMAND_HELP   AcpiGbl_DbCommandHelp[] =
236 {
237     {0, "\nGeneral-Purpose Commands:",          "\n"},
238     {1, "  Allocations",                       "Display list of current memory a
239     {2, "  Dump <Address>|<Namepath>",         "\n"},
240     {0, "       [Byte|Word|Dword|Qword]",      "Display ACPI objects or memory\n
241     {1, "  EnableAcpi",                        "Enable ACPI (hardware) mode\n"},
242     {1, "  Handlers",                          "Info about global handlers\n"},
243     {1, "  Help [Command]",                    "This help screen or individual c
244     {1, "  History",                           "Display command history buffer\n
245     {1, "  Level <DebugLevel>] [console]",     "Get/Set debug level for file or
246     {1, "  Locks",                             "Current status of internal mutex
247     {1, "  Osi [Install|Remove <name>]",       "Display or modify global _OSI li
248     {1, "  Quit or Exit",                      "Exit this command\n"},
249     {9, "  Stats [Allocations|Memory|Misc|",   "\n"},
250     {1, "      Objects|Sizes|Stack|Tables]",   "Display namespace and memory sta
251     {1, "     Allocations",                    "Display list of current memory a
252     {1, "     Memory",                         "Dump internal memory lists\n"},
253     {1, "     Misc",                           "Namespace search and mutex stats
254     {1, "     Objects",                        "Summary of namespace objects\n"}
255     {1, "     Sizes",                          "Sizes for each of the internal o
256     {1, "     Stack",                          "Display CPU stack usage\n"},
```

```
257     {1, "     Tables",                          "Info about current ACPI table(s)
258     {1, "  Tables",                             "Display info about loaded ACPI t
259     {1, "  Unload <Namepath>",                  "Unload an ACPI table via namespa
260     {1, "  ! <CommandNumber>",                  "Execute command from history buf
261     {1, "  !!",                                 "Execute last command again\n"},

263     {0, "\nNamespace Access Commands:",         "\n"},
264     {1, "  Businfo",                            "Display system bus info\n"},
265     {1, "  Disassemble <Method>",               "Disassemble a control method\n"}
266     {1, "  Find <AcpiName> (? is wildcard)",    "Find ACPI name(s) with wildcards
267     {1, "  Integrity",                          "Validate namespace integrity\n"}
268     {1, "  Methods",                            "Display list of loaded control m
269     {1, "  Namespace [Object] [Depth]",         "Display loaded namespace tree/su
270     {1, "  Notify <Object> <Value>",            "Send a notification on Object\n"
271     {1, "  Objects <ObjectType>",               "Display all objects of the given
272     {1, "  Owner <OwnerId> [Depth]",            "Display loaded namespace by obje
273     {1, "  Paths",                              "Display full pathnames of namesp
274     {1, "  Predefined",                         "Check all predefined names\n"},
275     {1, "  Prefix [<NamePath>]",                "Set or Get current execution pre
276     {1, "  References <Addr>",                  "Find all references to object at
277     {1, "  Resources [DeviceName]",             "Display Device resources (no arg
278     {1, "  Set N <NamedObject> <Value>",        "Set value for named integer\n"},
279     {1, "  Template <Object>",                  "Format/dump a Buffer/ResourceTem
280     {1, "  Terminate",                          "Delete namespace and all interna
281     {1, "  Type <Object>",                      "Display object type\n"},

283     {0, "\nControl Method Execution Commands:","\n"},
284     {1, "  Arguments (or Args)",                "Display method arguments\n"},
285     {1, "  Breakpoint <AmlOffset>",             "Set an AML execution breakpoint\
286     {1, "  Call",                               "Run to next control method invoc
287     {1, "  Debug <Namepath> [Arguments]",       "Single Step a control method\n"}
288     {6, "  Evaluate",                           "Synonym for Execute\n"},
289     {5, "  Execute <Namepath> [Arguments]",     "Execute control method\n"},
290     {1, "     Hex Integer",                     "Integer method argument\n"},
291     {1, "     \"Ascii String\"",                "String method argument\n"},
292     {1, "     (Hex Byte List)",                 "Buffer method argument\n"},
293     {1, "     [Package Element List]",          "Package method argument\n"},
294     {1, "  Go",                                 "Allow method to run to completio
295     {1, "  Information",                        "Display info about the current m
296     {1, "  Into",                               "Step into (not over) a method ca
297     {1, "  List [# of Aml Opcodes]",            "Display method ASL statements\n"
298     {1, "  Locals",                             "Display method local variables\n
299     {1, "  Results",                            "Display method result stack\n"},
300     {1, "  Set <A|L> <#> <Value>",              "Set method data (Arguments/Local
301     {1, "  Stop",                               "Terminate control method\n"},
302     {1, "  Thread <Threads><Loops><NamePath>", "Spawn threads to execute method(
303     {1, "  Trace <method name>",                "Trace method execution\n"},
304     {1, "  Tree",                               "Display control method calling t
305     {1, "  <Enter>",                            "Single step next AML opcode (ove

307     {0, "\nHardware Related Commands:",          "\n"},
308     {1, "  Event <F|G> <Value>",                "Generate AcpiEvent (Fixed/GPE)\n
309     {1, "  Gpe <GpeNum> <GpeBlock>",            "Simulate a GPE\n"},
310     {1, "  Gpes",                               "Display info on all GPEs\n"},
311     {1, "  Sci",                                "Generate an SCI\n"},
312     {1, "  Sleep [SleepState]",                 "Simulate sleep/wake sequence(s)

314     {0, "\nFile I/O Commands:",                  "\n"},
315     {1, "  Close",                              "Close debug output file\n"},
316     {1, "  Load <Input Filename>",              "Load ACPI table from a file\n"},
317     {1, "  Open <Output Filename>",             "Open a file for debug output\n"}
318     {0, NULL, NULL}
319 };


322 /******************************************************************************
```



```
323  *
324  * FUNCTION:    AcpiDbMatchCommandHelp
325  *
326  * PARAMETERS:  Command             - Command string to match
327  *              Help                - Help table entry to attempt match
328  *
329  * RETURN:      TRUE if command matched, FALSE otherwise
330  *
331  * DESCRIPTION: Attempt to match a command in the help table in order to
332  *              print help information for a single command.
333  *
334  *****************************************************************************/

336 static BOOLEAN
337 AcpiDbMatchCommandHelp (
338     char                        *Command,
339     const ACPI_DB_COMMAND_HELP  *Help)
340 {
341     char                    *Invocation = Help->Invocation;
342     UINT32                  LineCount;


345     /* Valid commands in the help table begin with a couple of spaces */

347     if (*Invocation != ' ')
348     {
349         return (FALSE);
350     }

352     while (*Invocation == ' ')
353     {
354         Invocation++;
355     }

357     /* Match command name (full command or substring) */

359     while ((*Command) && (*Invocation) && (*Invocation != ' '))
360     {
361         if (ACPI_TOLOWER (*Command) != ACPI_TOLOWER (*Invocation))
362         {
363             return (FALSE);
364         }

366         Invocation++;
367         Command++;
368     }

370     /* Print the appropriate number of help lines */

372     LineCount = Help->LineCount;
373     while (LineCount)
374     {
375         AcpiOsPrintf ("%-38s : %s", Help->Invocation, Help->Description);
376         Help++;
377         LineCount--;
378     }

380     return (TRUE);
381 }


384 /******************************************************************************
385  *
386  * FUNCTION:    AcpiDbDisplayCommandInfo
387  *
388  * PARAMETERS:  Command             - Command string to match
```

```
 *              DisplayAll          - Display all matching commands, or just
 *                                    the first one (substring match)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display help information for a Debugger command.
 *
 ******************************************************************************/

static void
AcpiDbDisplayCommandInfo (
    char                    *Command,
    BOOLEAN                 DisplayAll)
{
    const ACPI_DB_COMMAND_HELP  *Next;
    BOOLEAN                     Matched;


    Next = AcpiGbl_DbCommandHelp;
    while (Next->Invocation)
    {
        Matched = AcpiDbMatchCommandHelp (Command, Next);
        if (!DisplayAll && Matched)
        {
            return;
        }

        Next++;
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiDbDisplayHelp
 *
 * PARAMETERS:  Command             - Optional command string to display help.
 *                                    if not specified, all debugger command
 *                                    help strings are displayed
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display help for a single debugger command, or all of them.
 *
 ******************************************************************************/

static void
AcpiDbDisplayHelp (
    char                    *Command)
{
    const ACPI_DB_COMMAND_HELP  *Next = AcpiGbl_DbCommandHelp;


    if (!Command)
    {
        /* No argument to help, display help for all commands */

        while (Next->Invocation)
        {
            AcpiOsPrintf ("%-38s%s", Next->Invocation, Next->Description);
            Next++;
        }
    }
    else
    {
        /* Display help for all commands that match the subtring */
```



```
        AcpiDbDisplayCommandInfo (Command, TRUE);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiDbGetNextToken
 *
 * PARAMETERS:  String          - Command buffer
 *              Next            - Return value, end of next token
 *
 * RETURN:      Pointer to the start of the next token.
 *
 * DESCRIPTION: Command line parsing. Get the next token on the command line
 *
 ******************************************************************************/

char *
AcpiDbGetNextToken (
    char                    *String,
    char                    **Next,
    ACPI_OBJECT_TYPE        *ReturnType)
{
    char                    *Start;
    UINT32                  Depth;
    ACPI_OBJECT_TYPE        Type = ACPI_TYPE_INTEGER;


    /* At end of buffer? */

    if (!String || !(*String))
    {
        return (NULL);
    }

    /* Remove any spaces at the beginning */

    if (*String == ' ')
    {
        while (*String && (*String == ' '))
        {
            String++;
        }

        if (!(*String))
        {
            return (NULL);
        }
    }

    switch (*String)
    {
    case '"':

        /* This is a quoted string, scan until closing quote */

        String++;
        Start = String;
        Type = ACPI_TYPE_STRING;

        /* Find end of string */

        while (*String && (*String != '"'))
        {
```

```
521             String++;
522         }
523         break;

525     case '(':

527         /* This is the start of a buffer, scan until closing paren */

529         String++;
530         Start = String;
531         Type = ACPI_TYPE_BUFFER;

533         /* Find end of buffer */

535         while (*String && (*String != ')'))
536         {
537             String++;
538         }
539         break;

541     case '[':

543         /* This is the start of a package, scan until closing bracket */

545         String++;
546         Depth = 1;
547         Start = String;
548         Type = ACPI_TYPE_PACKAGE;

550         /* Find end of package (closing bracket) */

552         while (*String)
553         {
554             /* Handle String package elements */

556             if (*String == '"')
557             {
558                 /* Find end of string */

560                 String++;
561                 while (*String && (*String != '"'))
562                 {
563                     String++;
564                 }
565                 if (!(*String))
566                 {
567                     break;
568                 }
569             }
570             else if (*String == '[')
571             {
572                 Depth++;         /* A nested package declaration */
573             }
574             else if (*String == ']')
575             {
576                 Depth--;
577                 if (Depth == 0) /* Found final package closing bracket */
578                 {
579                     break;
580                 }
581             }

583             String++;
584         }
585         break;
```



```
587     default:

589         Start = String;

591         /* Find end of token */

593         while (*String && (*String != ' '))
594         {
595             String++;
596         }
597         break;
598     }

600     if (!(*String))
601     {
602         *Next = NULL;
603     }
604     else
605     {
606         *String = 0;
607         *Next = String + 1;
608     }

610     *ReturnType = Type;
611     return (Start);
612 }


615 /*******************************************************************************
616  *
617  * FUNCTION:    AcpiDbGetLine
618  *
619  * PARAMETERS:  InputBuffer         - Command line buffer
620  *
621  * RETURN:      Count of arguments to the command
622  *
623  * DESCRIPTION: Get the next command line from the user. Gets entire line
624  *              up to the next newline
625  *
626  ******************************************************************************/

628 static UINT32
629 AcpiDbGetLine (
630     char                    *InputBuffer)
631 {
632     UINT32                  i;
633     UINT32                  Count;
634     char                    *Next;
635     char                    *This;


638     if (AcpiUtSafeStrcpy (AcpiGbl_DbParsedBuf, sizeof (AcpiGbl_DbParsedBuf),
639         InputBuffer))
640     {
641         AcpiOsPrintf ("Buffer overflow while parsing input line (max %u characte
642             sizeof (AcpiGbl_DbParsedBuf));
643         return (0);
644     }

646     This = AcpiGbl_DbParsedBuf;
647     for (i = 0; i < ACPI_DEBUGGER_MAX_ARGS; i++)
648     {
649         AcpiGbl_DbArgs[i] = AcpiDbGetNextToken (This, &Next,
650             &AcpiGbl_DbArgTypes[i]);
651         if (!AcpiGbl_DbArgs[i])
652         {
```

```
653             break;
654         }

656         This = Next;
657     }

659     /* Uppercase the actual command */

661     if (AcpiGbl_DbArgs[0])
662     {
663         AcpiUtStrupr (AcpiGbl_DbArgs[0]);
664     }

666     Count = i;
667     if (Count)
668     {
669         Count--;  /* Number of args only */
670     }

672     return (Count);
673 }


676 /******************************************************************************
677  *
678  * FUNCTION:    AcpiDbMatchCommand
679  *
680  * PARAMETERS:  UserCommand             - User command line
681  *
682  * RETURN:      Index into command array, -1 if not found
683  *
684  * DESCRIPTION: Search command array for a command match
685  *
686  *****************************************************************************/

688 static UINT32
689 AcpiDbMatchCommand (
690     char                    *UserCommand)
691 {
692     UINT32                  i;


695     if (!UserCommand || UserCommand[0] == 0)
696     {
697         return (CMD_NULL);
698     }

700     for (i = CMD_FIRST_VALID; AcpiGbl_DbCommands[i].Name; i++)
701     {
702         if (ACPI_STRSTR (AcpiGbl_DbCommands[i].Name, UserCommand) ==
703                         AcpiGbl_DbCommands[i].Name)
704         {
705             return (i);
706         }
707     }

709     /* Command not recognized */

711     return (CMD_NOT_FOUND);
712 }


715 /******************************************************************************
716  *
717  * FUNCTION:    AcpiDbCommandDispatch
718  *
```



```
719  * PARAMETERS:  InputBuffer         - Command line buffer
720  *              WalkState           - Current walk
721  *              Op                  - Current (executing) parse op
722  *
723  * RETURN:      Status
724  *
725  * DESCRIPTION: Command dispatcher.
726  *
727  *****************************************************************************/

729 ACPI_STATUS
730 AcpiDbCommandDispatch (
731     char                    *InputBuffer,
732     ACPI_WALK_STATE         *WalkState,
733     ACPI_PARSE_OBJECT       *Op)
734 {
735     UINT32                  Temp;
736     UINT32                  CommandIndex;
737     UINT32                  ParamCount;
738     char                    *CommandLine;
739     ACPI_STATUS             Status = AE_CTRL_TRUE;


742     /* If AcpiTerminate has been called, terminate this thread */

744     if (AcpiGbl_DbTerminateThreads)
745     {
746         return (AE_CTRL_TERMINATE);
747     }


750     /* Add all commands that come here to the history buffer */

752     AcpiDbAddToHistory (InputBuffer);

754     ParamCount = AcpiDbGetLine (InputBuffer);
755     CommandIndex = AcpiDbMatchCommand (AcpiGbl_DbArgs[0]);
756     Temp = 0;

758     /* Verify that we have the minimum number of params */

760     if (ParamCount < AcpiGbl_DbCommands[CommandIndex].MinArgs)
761     {
762         AcpiOsPrintf ("%u parameters entered, [%s] requires %u parameters\n",
763             ParamCount, AcpiGbl_DbCommands[CommandIndex].Name,
764             AcpiGbl_DbCommands[CommandIndex].MinArgs);

766         AcpiDbDisplayCommandInfo (AcpiGbl_DbCommands[CommandIndex].Name, FALSE);
767         return (AE_CTRL_TRUE);
768     }

770     /* Decode and dispatch the command */

772     switch (CommandIndex)
773     {
774     case CMD_NULL:

776         if (Op)
777         {
778             return (AE_OK);
779         }
780         break;

782     case CMD_ALLOCATIONS:

784 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
```

```
785         AcpiUtDumpAllocations ((UINT32) -1, NULL);
786 #endif
787         break;

789     case CMD_ARGS:
790     case CMD_ARGUMENTS:

792         AcpiDbDisplayArguments ();
793         break;

795     case CMD_BATCH:

797         AcpiDbBatchExecute (AcpiGbl_DbArgs[1]);
798         break;

800     case CMD_BREAKPOINT:

802         AcpiDbSetMethodBreakpoint (AcpiGbl_DbArgs[1], WalkState, Op);
803         break;

805     case CMD_BUSINFO:

807         AcpiDbGetBusInfo ();
808         break;

810     case CMD_CALL:

812         AcpiDbSetMethodCallBreakpoint (Op);
813         Status = AE_OK;
814         break;

816     case CMD_CLOSE:

818         AcpiDbCloseDebugFile ();
819         break;

821     case CMD_DEBUG:

823         AcpiDbExecute (AcpiGbl_DbArgs[1],
824             &AcpiGbl_DbArgs[2], &AcpiGbl_DbArgTypes[2], EX_SINGLE_STEP);
825         break;

827     case CMD_DISASSEMBLE:
828     case CMD_DISASM:

830         (void) AcpiDbDisassembleMethod (AcpiGbl_DbArgs[1]);
831         break;

833     case CMD_DUMP:

835         AcpiDbDecodeAndDisplayObject (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
836         break;

838     case CMD_ENABLEACPI:
839 #if (!ACPI_REDUCED_HARDWARE)

841         Status = AcpiEnable();
842         if (ACPI_FAILURE(Status))
843         {
844             AcpiOsPrintf("AcpiEnable failed (Status=%X)\n", Status);
845             return (Status);
846         }
847 #endif /* !ACPI_REDUCED_HARDWARE */
848         break;

850     case CMD_EVENT:
```



```
852         AcpiOsPrintf ("Event command not implemented\n");
853         break;

855     case CMD_EVALUATE:
856     case CMD_EXECUTE:

858         AcpiDbExecute (AcpiGbl_DbArgs[1],
859             &AcpiGbl_DbArgs[2], &AcpiGbl_DbArgTypes[2], EX_NO_SINGLE_STEP);
860         break;

862     case CMD_FIND:

864         Status = AcpiDbFindNameInNamespace (AcpiGbl_DbArgs[1]);
865         break;

867     case CMD_GO:

869         AcpiGbl_CmSingleStep = FALSE;
870         return (AE_OK);

872     case CMD_GPE:

874         AcpiDbGenerateGpe (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
875         break;

877     case CMD_GPES:

879         AcpiDbDisplayGpes ();
880         break;

882     case CMD_HANDLERS:

884         AcpiDbDisplayHandlers ();
885         break;

887     case CMD_HELP:
888     case CMD_HELP2:

890         AcpiDbDisplayHelp (AcpiGbl_DbArgs[1]);
891         break;

893     case CMD_HISTORY:

895         AcpiDbDisplayHistory ();
896         break;

898     case CMD_HISTORY_EXE: /* ! command */

900         CommandLine = AcpiDbGetFromHistory (AcpiGbl_DbArgs[1]);
901         if (!CommandLine)
902         {
903             return (AE_CTRL_TRUE);
904         }

906         Status = AcpiDbCommandDispatch (CommandLine, WalkState, Op);
907         return (Status);

909     case CMD_HISTORY_LAST: /* !! command */

911         CommandLine = AcpiDbGetFromHistory (NULL);
912         if (!CommandLine)
913         {
914             return (AE_CTRL_TRUE);
915         }
```

```
917         Status = AcpiDbCommandDispatch (CommandLine, WalkState, Op);
918         return (Status);
919
920     case CMD_INFORMATION:
921
922         AcpiDbDisplayMethodInfo (Op);
923         break;
924
925     case CMD_INTEGRITY:
926
927         AcpiDbCheckIntegrity ();
928         break;
929
930     case CMD_INTO:
931
932         if (Op)
933         {
934             AcpiGbl_CmSingleStep = TRUE;
935             return (AE_OK);
936         }
937         break;
938
939     case CMD_LEVEL:
940
941         if (ParamCount == 0)
942         {
943             AcpiOsPrintf ("Current debug level for file output is:    %8.8lX\n",
944                 AcpiGbl_DbDebugLevel);
945             AcpiOsPrintf ("Current debug level for console output is: %8.8lX\n",
946                 AcpiGbl_DbConsoleDebugLevel);
947         }
948         else if (ParamCount == 2)
949         {
950             Temp = AcpiGbl_DbConsoleDebugLevel;
951             AcpiGbl_DbConsoleDebugLevel = ACPI_STRTOUL (AcpiGbl_DbArgs[1],
952                                         NULL, 16);
953             AcpiOsPrintf (
954                 "Debug Level for console output was %8.8lX, now %8.8lX\n",
955                 Temp, AcpiGbl_DbConsoleDebugLevel);
956         }
957         else
958         {
959             Temp = AcpiGbl_DbDebugLevel;
960             AcpiGbl_DbDebugLevel = ACPI_STRTOUL (AcpiGbl_DbArgs[1], NULL, 16);
961             AcpiOsPrintf (
962                 "Debug Level for file output was %8.8lX, now %8.8lX\n",
963                 Temp, AcpiGbl_DbDebugLevel);
964         }
965         break;
966
967     case CMD_LIST:
968
969         AcpiDbDisassembleAml (AcpiGbl_DbArgs[1], Op);
970         break;
971
972     case CMD_LOAD:
973
974         Status = AcpiDbGetTableFromFile (AcpiGbl_DbArgs[1], NULL);
975         break;
976
977     case CMD_LOCKS:
978
979         AcpiDbDisplayLocks ();
980         break;
981
982     case CMD_LOCALS:
```



```
984         AcpiDbDisplayLocals ();
985         break;
986
987     case CMD_METHODS:
988
989         Status = AcpiDbDisplayObjects ("METHOD", AcpiGbl_DbArgs[1]);
990         break;
991
992     case CMD_NAMESPACE:
993
994         AcpiDbDumpNamespace (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
995         break;
996
997     case CMD_NOTIFY:
998
999         Temp = ACPI_STRTOUL (AcpiGbl_DbArgs[2], NULL, 0);
1000        AcpiDbSendNotify (AcpiGbl_DbArgs[1], Temp);
1001        break;
1002
1003    case CMD_OBJECTS:
1004
1005        AcpiUtStrupr (AcpiGbl_DbArgs[1]);
1006        Status = AcpiDbDisplayObjects (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1007        break;
1008
1009    case CMD_OPEN:
1010
1011        AcpiDbOpenDebugFile (AcpiGbl_DbArgs[1]);
1012        break;
1013
1014    case CMD_OSI:
1015
1016        AcpiDbDisplayInterfaces (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1017        break;
1018
1019    case CMD_OWNER:
1020
1021        AcpiDbDumpNamespaceByOwner (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1022        break;
1023
1024    case CMD_PATHS:
1025
1026        AcpiDbDumpNamespacePaths ();
1027        break;
1028
1029    case CMD_PREDEFINED:
1030
1031        AcpiDbCheckPredefinedNames ();
1032        break;
1033
1034    case CMD_PREFIX:
1035
1036        AcpiDbSetScope (AcpiGbl_DbArgs[1]);
1037        break;
1038
1039    case CMD_REFERENCES:
1040
1041        AcpiDbFindReferences (AcpiGbl_DbArgs[1]);
1042        break;
1043
1044    case CMD_RESOURCES:
1045
1046        AcpiDbDisplayResources (AcpiGbl_DbArgs[1]);
1047        break;
```

```
1049     case CMD_RESULTS:

1051         AcpiDbDisplayResults ();
1052         break;

1054     case CMD_SCI:

1056         AcpiDbGenerateSci ();
1057         break;

1059     case CMD_SET:

1061         AcpiDbSetMethodData (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2],
1062             AcpiGbl_DbArgs[3]);
1063         break;

1065     case CMD_SLEEP:

1067         Status = AcpiDbSleep (AcpiGbl_DbArgs[1]);
1068         break;

1070     case CMD_STATS:

1072         Status = AcpiDbDisplayStatistics (AcpiGbl_DbArgs[1]);
1073         break;

1075     case CMD_STOP:

1077         return (AE_NOT_IMPLEMENTED);

1079     case CMD_TABLES:

1081         AcpiDbDisplayTableInfo (AcpiGbl_DbArgs[1]);
1082         break;

1084     case CMD_TEMPLATE:

1086         AcpiDbDisplayTemplate (AcpiGbl_DbArgs[1]);
1087         break;

1089     case CMD_TERMINATE:

1091         AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
1092         AcpiUtSubsystemShutdown ();

1094         /*
1095          * TBD: [Restructure] Need some way to re-initialize without
1096          * re-creating the semaphores!
1097          */

1099         /*  AcpiInitialize (NULL);  */
1100         break;

1102     case CMD_THREADS:

1104         AcpiDbCreateExecutionThreads (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2],
1105             AcpiGbl_DbArgs[3]);
1106         break;

1108     case CMD_TRACE:

1110         (void) AcpiDebugTrace (AcpiGbl_DbArgs[1],0,0,1);
1111         break;

1113     case CMD_TREE:
```



```
1115         AcpiDbDisplayCallingTree ();
1116         break;

1118     case CMD_TYPE:

1120         AcpiDbDisplayObjectType (AcpiGbl_DbArgs[1]);
1121         break;

1123     case CMD_UNLOAD:

1125         AcpiDbUnloadAcpiTable (AcpiGbl_DbArgs[1]);
1126         break;

1128     case CMD_EXIT:
1129     case CMD_QUIT:

1131         if (Op)
1132         {
1133             AcpiOsPrintf ("Method execution terminated\n");
1134             return (AE_CTRL_TERMINATE);
1135         }

1137         if (!AcpiGbl_DbOutputToFile)
1138         {
1139             AcpiDbgLevel = ACPI_DEBUG_DEFAULT;
1140         }

1142         AcpiDbCloseDebugFile ();
1143         AcpiGbl_DbTerminateThreads = TRUE;
1144         return (AE_CTRL_TERMINATE);

1146     case CMD_NOT_FOUND:
1147     default:

1149         AcpiOsPrintf ("%s: unknown command\n", AcpiGbl_DbArgs[0]);
1150         return (AE_CTRL_TRUE);
1151     }

1153     if (ACPI_SUCCESS (Status))
1154     {
1155         Status = AE_CTRL_TRUE;
1156     }

1158     return (Status);
1159 }


1162 /******************************************************************************
1163  *
1164  * FUNCTION:    AcpiDbExecuteThread
1165  *
1166  * PARAMETERS:  Context         - Not used
1167  *
1168  * RETURN:      None
1169  *
1170  * DESCRIPTION: Debugger execute thread. Waits for a command line, then
1171  *              simply dispatches it.
1172  *
1173  *****************************************************************************/

1175 void ACPI_SYSTEM_XFACE
1176 AcpiDbExecuteThread (
1177     void                    *Context)
1178 {
1179     ACPI_STATUS             Status = AE_OK;
1180     ACPI_STATUS             MStatus;
```

```
1183     while (Status != AE_CTRL_TERMINATE)
1184     {
1185         AcpiGbl_MethodExecuting = FALSE;
1186         AcpiGbl_StepToNextCall = FALSE;

1188         MStatus = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_READY);
1189         if (ACPI_FAILURE (MStatus))
1190         {
1191             return;
1192         }

1194         Status = AcpiDbCommandDispatch (AcpiGbl_DbLineBuf, NULL, NULL);

1196         MStatus = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
1197         if (ACPI_FAILURE (MStatus))
1198         {
1199             return;
1200         }
1201     }
1202 }


1205 /*******************************************************************************
1206  *
1207  * FUNCTION:    AcpiDbSingleThread
1208  *
1209  * PARAMETERS:  None
1210  *
1211  * RETURN:      None
1212  *
1213  * DESCRIPTION: Debugger execute thread. Waits for a command line, then
1214  *              simply dispatches it.
1215  *
1216  ******************************************************************************/

1218 static void
1219 AcpiDbSingleThread (
1220     void)
1221 {

1223     AcpiGbl_MethodExecuting = FALSE;
1224     AcpiGbl_StepToNextCall = FALSE;

1226     (void) AcpiDbCommandDispatch (AcpiGbl_DbLineBuf, NULL, NULL);
1227 }


1230 /*******************************************************************************
1231  *
1232  * FUNCTION:    AcpiDbUserCommands
1233  *
1234  * PARAMETERS:  Prompt              - User prompt (depends on mode)
1235  *              Op                  - Current executing parse op
1236  *
1237  * RETURN:      None
1238  *
1239  * DESCRIPTION: Command line execution for the AML debugger. Commands are
1240  *              matched and dispatched here.
1241  *
1242  ******************************************************************************/

1244 ACPI_STATUS
1245 AcpiDbUserCommands (
1246     char                    Prompt,
```



```
1247     ACPI_PARSE_OBJECT       *Op)
1248 {
1249     ACPI_STATUS             Status = AE_OK;


1252     AcpiOsPrintf ("\n");

1254     /* TBD: [Restructure] Need a separate command line buffer for step mode */

1256     while (!AcpiGbl_DbTerminateThreads)
1257     {
1258         /* Force output to console until a command is entered */

1260         AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);

1262         /* Different prompt if method is executing */

1264         if (!AcpiGbl_MethodExecuting)
1265         {
1266             AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_COMMAND_PROMPT);
1267         }
1268         else
1269         {
1270             AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_EXECUTE_PROMPT);
1271         }

1273         /* Get the user input line */

1275         Status = AcpiOsGetLine (AcpiGbl_DbLineBuf,
1276             ACPI_DB_LINE_BUFFER_SIZE, NULL);
1277         if (ACPI_FAILURE (Status))
1278         {
1279             ACPI_EXCEPTION ((AE_INFO, Status, "While parsing command line"));
1280             return (Status);
1281         }

1283         /* Check for single or multithreaded debug */

1285         if (AcpiGbl_DebuggerConfiguration & DEBUGGER_MULTI_THREADED)
1286         {
1287             /*
1288              * Signal the debug thread that we have a command to execute,
1289              * and wait for the command to complete.
1290              */
1291             Status = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_READY);
1292             if (ACPI_FAILURE (Status))
1293             {
1294                 return (Status);
1295             }

1297             Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
1298             if (ACPI_FAILURE (Status))
1299             {
1300                 return (Status);
1301             }
1302         }
1303         else
1304         {
1305             /* Just call to the command line interpreter */

1307             AcpiDbSingleThread ();
1308         }
1309     }

1311     /*
1312      * Only this thread (the original thread) should actually terminate the
```

```
1313      * subsystem, because all the semaphores are deleted during termination
1314      */
1315     Status = AcpiTerminate ();
1316     return (Status);
1317 }

1319 #endif  /* ACPI_DEBUGGER */
```

****
16409 Thu Dec 26 13:48:43 2013
new/usr/src/common/acpica/components/debugger/dbmethod.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dbmethod - Debug commands for control methods
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"
#include "acnamesp.h"
#include "acdebug.h"
#include "acdisasm.h"
#include "acparser.h"
#include "acpredef.h"


#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbmethod")
```



```
/* Local prototypes */

static ACPI_STATUS
AcpiDbWalkForExecute (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    AcpiDbSetMethodBreakpoint
 *
 * PARAMETERS:  Location            - AML offset of breakpoint
 *              WalkState           - Current walk info
 *              Op                  - Current Op (from parse walk)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Set a breakpoint in a control method at the specified
 *              AML offset
 *
 *****************************************************************************/

void
AcpiDbSetMethodBreakpoint (
    char                    *Location,
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    UINT32                  Address;


    if (!Op)
    {
        AcpiOsPrintf ("There is no method currently executing\n");
        return;
    }

    /* Get and verify the breakpoint address */

    Address = ACPI_STRTOUL (Location, NULL, 16);
    if (Address <= Op->Common.AmlOffset)
    {
        AcpiOsPrintf ("Breakpoint %X is beyond current address %X\n",
            Address, Op->Common.AmlOffset);
    }

    /* Save breakpoint in current walk */

    WalkState->UserBreakpoint = Address;
    AcpiOsPrintf ("Breakpoint set at AML offset %X\n", Address);
}
```

_____unchanged_portion_omitted_

```
/******************************************************************************
 *
 * FUNCTION:    AcpiDbSetMethodData
 *
 * PARAMETERS:  TypeArg         - L for local, A for argument
 *              IndexArg        - which one
 *              ValueArg        - Value to set.
 *
```

```
 154  * RETURN:      None
 155  *
 156  * DESCRIPTION: Set a local or argument for the running control method.
 157  *              NOTE: only object supported is Number.
 158  *
 159  ******************************************************************************/

 161 void
 162 AcpiDbSetMethodData (
 163     char                    *TypeArg,
 164     char                    *IndexArg,
 165     char                    *ValueArg)
 166 {
 167     char                    Type;
 168     UINT32                  Index;
 169     UINT32                  Value;
 170     ACPI_WALK_STATE         *WalkState;
 171     ACPI_OPERAND_OBJECT     *ObjDesc;
 172     ACPI_STATUS             Status;
 173     ACPI_NAMESPACE_NODE     *Node;


 176     /* Validate TypeArg */

 178     AcpiUtStrupr (TypeArg);
 179     Type = TypeArg[0];
 180     if ((Type != 'L') &&
 181         (Type != 'A') &&
 182         (Type != 'N'))
 183     {
 184         AcpiOsPrintf ("Invalid SET operand: %s\n", TypeArg);
 185         return;
 186     }

 188     Value = ACPI_STRTOUL (ValueArg, NULL, 16);

 190     if (Type == 'N')
 191     {
 192         Node = AcpiDbConvertToNode (IndexArg);
 193         if (Node->Type != ACPI_TYPE_INTEGER)
 194         {
 195             AcpiOsPrintf ("Can only set Integer nodes\n");
 196             return;
 197         }
 198         ObjDesc = Node->Object;
 199         ObjDesc->Integer.Value = Value;
 200         return;
 201     }

 203     /* Get the index and value */

 205     Index = ACPI_STRTOUL (IndexArg, NULL, 16);

 207     WalkState = AcpiDsGetCurrentWalkState (AcpiGbl_CurrentWalkList);
 208     if (!WalkState)
 209     {
 210         AcpiOsPrintf ("There is no method currently executing\n");
 211         return;
 212     }

 214     /* Create and initialize the new object */

 216     ObjDesc = AcpiUtCreateIntegerObject ((UINT64) Value);
 217     if (!ObjDesc)
 218     {
 219         AcpiOsPrintf ("Could not create an internal object\n");
```



```
 220         return;
 221     }

 223     /* Store the new object into the target */

 225     switch (Type)
 226     {
 227     case 'A':

 229         /* Set a method argument */

 231         if (Index > ACPI_METHOD_MAX_ARG)
 232         {
 233             AcpiOsPrintf ("Arg%u - Invalid argument name\n", Index);
 234             goto Cleanup;
 235         }

 237         Status = AcpiDsStoreObjectToLocal (ACPI_REFCLASS_ARG, Index, ObjDesc,
 238                     WalkState);
 239         if (ACPI_FAILURE (Status))
 240         {
 241             goto Cleanup;
 242         }

 244         ObjDesc = WalkState->Arguments[Index].Object;

 246         AcpiOsPrintf ("Arg%u: ", Index);
 247         AcpiDmDisplayInternalObject (ObjDesc, WalkState);
 248         break;

 250     case 'L':

 252         /* Set a method local */

 254         if (Index > ACPI_METHOD_MAX_LOCAL)
 255         {
 256             AcpiOsPrintf ("Local%u - Invalid local variable name\n", Index);
 257             goto Cleanup;
 258         }

 260         Status = AcpiDsStoreObjectToLocal (ACPI_REFCLASS_LOCAL, Index, ObjDesc,
 261                     WalkState);
 262         if (ACPI_FAILURE (Status))
 263         {
 264             goto Cleanup;
 265         }

 267         ObjDesc = WalkState->LocalVariables[Index].Object;

 269         AcpiOsPrintf ("Local%u: ", Index);
 270         AcpiDmDisplayInternalObject (ObjDesc, WalkState);
 271         break;

 273     default:

 275         break;
 276     }

 278 Cleanup:
 279     AcpiUtRemoveReference (ObjDesc);
 280 }
_____unchanged_portion_omitted_


 320 /******************************************************************************
 321  *
```

```
 * FUNCTION:    AcpiDbDisassembleMethod
 *
 * PARAMETERS:  Name            - Name of control method
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display disassembled AML (ASL) starting from Op for the number
 *              of statements specified.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDbDisassembleMethod (
    char                    *Name)
{
    ACPI_STATUS             Status;
    ACPI_PARSE_OBJECT       *Op;
    ACPI_WALK_STATE         *WalkState;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_NAMESPACE_NODE     *Method;


    Method = AcpiDbConvertToNode (Name);
    if (!Method)
    {
        return (AE_BAD_PARAMETER);
    }

    if (Method->Type != ACPI_TYPE_METHOD)
    {
        ACPI_ERROR ((AE_INFO, "%s (%s): Object must be a control method",
            Name, AcpiUtGetTypeName (Method->Type)));
        return (AE_BAD_PARAMETER);
    }

    ObjDesc = Method->Object;

    Op = AcpiPsCreateScopeOp ();
    if (!Op)
    {
        return (AE_NO_MEMORY);
    }

    /* Create and initialize a new walk state */

    WalkState = AcpiDsCreateWalkState (0, Op, NULL, NULL);
    if (!WalkState)
    {
        return (AE_NO_MEMORY);
    }

    Status = AcpiDsInitAmlWalk (WalkState, Op, NULL,
        ObjDesc->Method.AmlStart,
        ObjDesc->Method.AmlLength, NULL, ACPI_IMODE_LOAD_PASS1);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    Status = AcpiUtAllocateOwnerId (&ObjDesc->Method.OwnerId);
    WalkState->OwnerId = ObjDesc->Method.OwnerId;
    /* Parse the AML */

    /* Push start scope on scope stack and make it current */

    Status = AcpiDsScopeStackPush (Method,
```



```
        Method->Type, WalkState);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Parse the entire method AML including deferred operators */

    WalkState->ParseFlags &= ~ACPI_PARSE_DELETE_TREE;
    WalkState->ParseFlags |= ACPI_PARSE_DISASSEMBLE;

    Status = AcpiPsParseAml (WalkState);
    (void) AcpiDmParseDeferredOps (Op);

    /* Now we can disassemble the method */

    AcpiGbl_DbOpt_verbose = FALSE;
    AcpiDmDisassemble (NULL, Op, 0);
    AcpiGbl_DbOpt_verbose = TRUE;

    AcpiPsDeleteParseTree (Op);

    /* Method cleanup */

    AcpiNsDeleteNamespaceSubtree (Method);
    AcpiNsDeleteNamespaceByOwner (ObjDesc->Method.OwnerId);
    AcpiUtReleaseOwnerId (&ObjDesc->Method.OwnerId);
    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbWalkForExecute
 *
 * PARAMETERS:  Callback from WalkNamespace
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Batch execution module. Currently only executes predefined
 *              ACPI names.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDbWalkForExecute (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue)
{
    ACPI_NAMESPACE_NODE         *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
    ACPI_DB_EXECUTE_WALK        *Info = (ACPI_DB_EXECUTE_WALK *) Context;
    ACPI_EXECUTE_WALK       *Info = (ACPI_EXECUTE_WALK *) Context;
    ACPI_BUFFER             ReturnObj;
    ACPI_STATUS             Status;
    char                        *Pathname;
    const ACPI_PREDEFINED_INFO  *Predefined;
    UINT32                  i;
    ACPI_DEVICE_INFO            *ObjInfo;
    ACPI_OBJECT_LIST            ParamObjects;
    ACPI_OBJECT                 Params[ACPI_METHOD_NUM_ARGS];
    ACPI_OBJECT                 *ThisParam;
    ACPI_BUFFER                 ReturnObj;
    ACPI_STATUS                 Status;
    UINT16                      ArgTypeList;
```

```
    UINT8                   ArgCount;
    UINT8                   ArgType;
    UINT32                  i;
    const ACPI_PREDEFINED_INFO *Predefined;


    /* The name must be a predefined ACPI name */

    Predefined = AcpiUtMatchPredefinedMethod (Node->Name.Ascii);
    Predefined = AcpiNsCheckForPredefinedName (Node);
    if (!Predefined)
    {
        return (AE_OK);
    }

    if (Node->Type == ACPI_TYPE_LOCAL_SCOPE)
    {
        return (AE_OK);
    }

    Pathname = AcpiNsGetExternalPathname (Node);
    if (!Pathname)
    {
        return (AE_OK);
    }

    /* Get the object info for number of method parameters */

    Status = AcpiGetObjectInfo (ObjHandle, &ObjInfo);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    ParamObjects.Count = 0;
    ParamObjects.Pointer = NULL;
    ParamObjects.Count   = 0;

    if (ObjInfo->Type == ACPI_TYPE_METHOD)
    {
        /* Setup default parameters (with proper types) */
        /* Setup default parameters */

        ArgTypeList = Predefined->Info.ArgumentList;
        ArgCount = METHOD_GET_ARG_COUNT (ArgTypeList);

        /*
         * Setup the ACPI-required number of arguments, regardless of what
         * the actual method defines. If there is a difference, then the
         * method is wrong and a warning will be issued during execution.
         */
        ThisParam = Params;
        for (i = 0; i < ArgCount; i++)
        for (i = 0; i < ObjInfo->ParamCount; i++)
        {
            ArgType = METHOD_GET_NEXT_TYPE (ArgTypeList);
            ThisParam->Type = ArgType;

            switch (ArgType)
            {
            case ACPI_TYPE_INTEGER:

                ThisParam->Integer.Value = 1;
                break;

            case ACPI_TYPE_STRING:
```



```
                ThisParam->String.Pointer = "This is the default argument string
                ThisParam->String.Length = ACPI_STRLEN (ThisParam->String.Pointe
                break;

            case ACPI_TYPE_BUFFER:

                ThisParam->Buffer.Pointer = (UINT8 *) Params; /* just a garbage
                ThisParam->Buffer.Length = 48;
                break;

            case ACPI_TYPE_PACKAGE:

                ThisParam->Package.Elements = NULL;
                ThisParam->Package.Count = 0;
                break;

            default:

                AcpiOsPrintf ("%s: Unsupported argument type: %u\n",
                    Pathname, ArgType);
                break;
            Params[i].Type           = ACPI_TYPE_INTEGER;
            Params[i].Integer.Value  = 1;
            }

            ThisParam++;
        }

        ParamObjects.Count = ArgCount;
        ParamObjects.Pointer = Params;
        ParamObjects.Count       = ObjInfo->ParamCount;
    }

    ACPI_FREE (ObjInfo);
    ReturnObj.Pointer = NULL;
    ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

    /* Do the actual method execution */

    AcpiGbl_MethodExecuting = TRUE;

    Status = AcpiEvaluateObject (Node, NULL, &ParamObjects, &ReturnObj);

    AcpiOsPrintf ("%-32s returned %s\n", Pathname, AcpiFormatException (Status))
    AcpiGbl_MethodExecuting = FALSE;
    ACPI_FREE (Pathname);

    /* Ignore status from method execution */

    Status = AE_OK;

    /* Update count, check if we have executed enough methods */

    Info->Count++;
    if (Info->Count >= Info->MaxCount)
    {
        Status = AE_CTRL_TERMINATE;
    }

    return (Status);
}


/*****************************************************************************
 *
```

```
 573  * FUNCTION:    AcpiDbBatchExecute
 574  *
 575  * PARAMETERS:  CountArg            - Max number of methods to execute
 576  *
 577  * RETURN:      None
 578  *
 579  * DESCRIPTION: Namespace batch execution. Execute predefined names in the
 580  *              namespace, up to the max count, if specified.
 581  *
 582  ******************************************************************************/

 584 void
 585 AcpiDbBatchExecute (
 586     char                    *CountArg)
 587 {
 588     ACPI_DB_EXECUTE_WALK    Info;
 505     ACPI_EXECUTE_WALK       Info;


 591     Info.Count = 0;
 592     Info.MaxCount = ACPI_UINT32_MAX;

 594     if (CountArg)
 595     {
 596         Info.MaxCount = ACPI_STRTOUL (CountArg, NULL, 0);
 597     }


 600     /* Search all nodes in namespace */

 602     (void) AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT, ACPI_UINT32_MAX,
 603                 AcpiDbWalkForExecute, NULL, (void *) &Info, NULL);

 605     AcpiOsPrintf ("Evaluated %u predefined names in the namespace\n", Info.Count
 522     AcpiOsPrintf ("Executed %u predefined names in the namespace\n", Info.Count)
 606 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   26977 Thu Dec 26 13:48:43 2013
new/usr/src/common/acpica/components/debugger/dbnames.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dbnames - Debugger commands for the acpi namespace
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #include "acpi.h"
  46 #include "accommon.h"
  47 #include "acnamesp.h"
  48 #include "acdebug.h"
  49 #include "acpredef.h"


  52 #ifdef ACPI_DEBUGGER

  54 #define _COMPONENT          ACPI_CA_DEBUGGER
  55         ACPI_MODULE_NAME    ("dbnames")


  58 /* Local prototypes */
```



```
  60 static ACPI_STATUS
  61 AcpiDbWalkAndMatchName (
  62     ACPI_HANDLE             ObjHandle,
  63     UINT32                  NestingLevel,
  64     void                    *Context,
  65     void                    **ReturnValue);

  67 static ACPI_STATUS
  68 AcpiDbWalkForPredefinedNames (
  69     ACPI_HANDLE             ObjHandle,
  70     UINT32                  NestingLevel,
  71     void                    *Context,
  72     void                    **ReturnValue);

  74 static ACPI_STATUS
  75 AcpiDbWalkForSpecificObjects (
  76     ACPI_HANDLE             ObjHandle,
  77     UINT32                  NestingLevel,
  78     void                    *Context,
  79     void                    **ReturnValue);

  81 static ACPI_STATUS
  82 AcpiDbIntegrityWalk (
  83     ACPI_HANDLE             ObjHandle,
  84     UINT32                  NestingLevel,
  85     void                    *Context,
  86     void                    **ReturnValue);

  88 static ACPI_STATUS
  89 AcpiDbWalkForReferences (
  90     ACPI_HANDLE             ObjHandle,
  91     UINT32                  NestingLevel,
  92     void                    *Context,
  93     void                    **ReturnValue);

  95 static ACPI_STATUS
  96 AcpiDbBusWalk (
  97     ACPI_HANDLE             ObjHandle,
  98     UINT32                  NestingLevel,
  99     void                    *Context,
 100     void                    **ReturnValue);

 102 /*
 103  * Arguments for the Objects command
 104  * These object types map directly to the ACPI_TYPES
 105  */
 106 static ACPI_DB_ARGUMENT_INFO    AcpiDbObjectTypes [] =
 105 static ARGUMENT_INFO        AcpiDbObjectTypes [] =
 107 {
 108     {"ANY"},
 109     {"INTEGERS"},
 110     {"STRINGS"},
 111     {"BUFFERS"},
 112     {"PACKAGES"},
 113     {"FIELDS"},
 114     {"DEVICES"},
 115     {"EVENTS"},
 116     {"METHODS"},
 117     {"MUTEXES"},
 118     {"REGIONS"},
 119     {"POWERRESOURCES"},
 120     {"PROCESSORS"},
 121     {"THERMALZONES"},
 122     {"BUFFERFIELDS"},
 123     {"DDBHANDLES"},
```

```
    {"DEBUG"},
    {"REGIONFIELDS"},
    {"BANKFIELDS"},
    {"INDEXFIELDS"},
    {"REFERENCES"},
    {"ALIAS"},
    {NULL}          /* Must be null terminated */
};


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbSetScope
 *
 * PARAMETERS:  Name                - New scope path
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Set the "current scope" as maintained by this utility.
 *              The scope is used as a prefix to ACPI paths.
 *
 ****************************************************************************/

void
AcpiDbSetScope (
    char                    *Name)
{
    ACPI_STATUS             Status;
    ACPI_NAMESPACE_NODE     *Node;


    if (!Name || Name[0] == 0)
    {
        AcpiOsPrintf ("Current scope: %s\n", AcpiGbl_DbScopeBuf);
        return;
    }

    AcpiDbPrepNamestring (Name);

    if (ACPI_IS_ROOT_PREFIX (Name[0]))
    if (Name[0] == '\')
    {
        /* Validate new scope from the root */

        Status = AcpiNsGetNode (AcpiGbl_RootNode, Name, ACPI_NS_NO_UPSEARCH,
                    &Node);
        if (ACPI_FAILURE (Status))
        {
            goto ErrorExit;
        }

        AcpiGbl_DbScopeBuf[0] = 0;
        ACPI_STRCPY (AcpiGbl_DbScopeBuf, Name);
        ACPI_STRCAT (AcpiGbl_DbScopeBuf, "\");
    }
    else
    {
        /* Validate new scope relative to old scope */

        Status = AcpiNsGetNode (AcpiGbl_DbScopeNode, Name, ACPI_NS_NO_UPSEARCH,
                    &Node);
        if (ACPI_FAILURE (Status))
        {
            goto ErrorExit;
        }
    }
```



```
    /* Build the final pathname */

    if (AcpiUtSafeStrcat (AcpiGbl_DbScopeBuf, sizeof (AcpiGbl_DbScopeBuf),
        Name))
    {
        Status = AE_BUFFER_OVERFLOW;
        goto ErrorExit;
        ACPI_STRCAT (AcpiGbl_DbScopeBuf, Name);
        ACPI_STRCAT (AcpiGbl_DbScopeBuf, "\");
    }

    if (AcpiUtSafeStrcat (AcpiGbl_DbScopeBuf, sizeof (AcpiGbl_DbScopeBuf),
        "\"))
    {
        Status = AE_BUFFER_OVERFLOW;
        goto ErrorExit;
    }

    AcpiGbl_DbScopeNode = Node;
    AcpiOsPrintf ("New scope: %s\n", AcpiGbl_DbScopeBuf);
    return;

ErrorExit:

    AcpiOsPrintf ("Could not attach scope: %s, %s\n",
        Name, AcpiFormatException (Status));
}
_____unchanged_portion_omitted_


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbDumpNamespacePaths
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump entire namespace with full object pathnames and object
 *              type information. Alternative to "namespace" command.
 *
 ****************************************************************************/

void
AcpiDbDumpNamespacePaths (
    void)
{

    AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
    AcpiOsPrintf ("ACPI Namespace (from root):\n");

    /* Display the entire namespace */

    AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
    AcpiNsDumpObjectPaths (ACPI_TYPE_ANY, ACPI_DISPLAY_SUMMARY,
        ACPI_UINT32_MAX, ACPI_OWNER_ID_MAX, AcpiGbl_RootNode);

    AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDbDumpNamespaceByOwner
 *
```

```
 304  * PARAMETERS:  OwnerArg        - Owner ID whose nodes will be displayed
 305  *              DepthArg        - Maximum tree depth to be dumped
 306  *
 307  * RETURN:      None
 308  *
 309  * DESCRIPTION: Dump elements of the namespace that are owned by the OwnerId.
 310  *
 311  ******************************************************************************/

 313 void
 314 AcpiDbDumpNamespaceByOwner (
 315     char                    *OwnerArg,
 316     char                    *DepthArg)
 317 {
 318     ACPI_HANDLE             SubtreeEntry = AcpiGbl_RootNode;
 319     UINT32                  MaxDepth = ACPI_UINT32_MAX;
 320     ACPI_OWNER_ID           OwnerId;


 323     OwnerId = (ACPI_OWNER_ID) ACPI_STRTOUL (OwnerArg, NULL, 0);

 325     /* Now we can check for the depth argument */

 327     if (DepthArg)
 328     {
 329         MaxDepth = ACPI_STRTOUL (DepthArg, NULL, 0);
 330     }

 332     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
 333     AcpiOsPrintf ("ACPI Namespace by owner %X:\n", OwnerId);

 335     /* Display the subtree */

 337     AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
 338     AcpiNsDumpObjects (ACPI_TYPE_ANY, ACPI_DISPLAY_SUMMARY, MaxDepth, OwnerId,
 339         SubtreeEntry);
 340     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
 341 }
```

*unchanged_portion_omitted_*

```
 456 /******************************************************************************
 457  *
 458  * FUNCTION:    AcpiDbWalkForPredefinedNames
 459  *
 460  * PARAMETERS:  Callback from WalkNamespace
 461  *
 462  * RETURN:      Status
 463  *
 464  * DESCRIPTION: Detect and display predefined ACPI names (names that start with
 465  *              an underscore)
 466  *
 467  *****************************************************************************/

 469 static ACPI_STATUS
 470 AcpiDbWalkForPredefinedNames (
 471     ACPI_HANDLE             ObjHandle,
 472     UINT32                  NestingLevel,
 473     void                    *Context,
 474     void                    **ReturnValue)
 475 {
 476     ACPI_NAMESPACE_NODE         *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
 477     UINT32                  *Count = (UINT32 *) Context;
 478     const ACPI_PREDEFINED_INFO  *Predefined;
 479     const ACPI_PREDEFINED_INFO  *Package = NULL;
 480     char                        *Pathname;
```



```
 481     char                        StringBuffer[48];


 484     Predefined = AcpiUtMatchPredefinedMethod (Node->Name.Ascii);
 439     Predefined = AcpiNsCheckForPredefinedName (Node);
 485     if (!Predefined)
 486     {
 487         return (AE_OK);
 488     }

 490     Pathname = AcpiNsGetExternalPathname (Node);
 491     if (!Pathname)
 492     {
 493         return (AE_OK);
 494     }

 496     /* If method returns a package, the info is in the next table entry */

 498     if (Predefined->Info.ExpectedBtypes & ACPI_RTYPE_PACKAGE)
 453     if (Predefined->Info.ExpectedBtypes & ACPI_BTYPE_PACKAGE)
 499     {
 500         Package = Predefined + 1;
 501     }

 503     AcpiUtGetExpectedReturnTypes (StringBuffer,
 504         Predefined->Info.ExpectedBtypes);
 458     AcpiOsPrintf ("%-32s arg %X ret %2.2X", Pathname,
 459         Predefined->Info.ParamCount, Predefined->Info.ExpectedBtypes);

 506     AcpiOsPrintf ("%-32s Arguments %X, Return Types: %s", Pathname,
 507         METHOD_GET_ARG_COUNT (Predefined->Info.ArgumentList),
 508         StringBuffer);

 510     if (Package)
 511     {
 512         AcpiOsPrintf (" (PkgType %2.2X, ObjType %2.2X, Count %2.2X)",
 463         AcpiOsPrintf (" PkgType %2.2X ObjType %2.2X Count %2.2X",
 513             Package->RetInfo.Type, Package->RetInfo.ObjectType1,
 514             Package->RetInfo.Count1);
 515     }

 517     AcpiOsPrintf("\n");

 519     /* Check that the declared argument count matches the ACPI spec */

 521     AcpiNsCheckAcpiCompliance (Pathname, Node, Predefined);

 470     AcpiNsCheckParameterCount (Pathname, Node, ACPI_UINT32_MAX, Predefined);
 523     ACPI_FREE (Pathname);
 524     (*Count)++;

 525     return (AE_OK);
 526 }
```

*unchanged_portion_omitted_*

```
 660 /******************************************************************************
 661  *
 662  * FUNCTION:    AcpiDbIntegrityWalk
 663  *
 664  * PARAMETERS:  Callback from WalkNamespace
 665  *
 666  * RETURN:      Status
 667  *
 668  * DESCRIPTION: Examine one NS node for valid values.
 669  *
```

```
 670  *****************************************************************************/

 672 static ACPI_STATUS
 673 AcpiDbIntegrityWalk (
 674     ACPI_HANDLE             ObjHandle,
 675     UINT32                  NestingLevel,
 676     void                    *Context,
 677     void                    **ReturnValue)
 678 {
 679     ACPI_INTEGRITY_INFO     *Info = (ACPI_INTEGRITY_INFO *) Context;
 680     ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
 681     ACPI_OPERAND_OBJECT     *Object;
 682     BOOLEAN                 Alias = TRUE;


 685     Info->Nodes++;

 687     /* Verify the NS node, and dereference aliases */

 689     while (Alias)
 690     {
 691         if (ACPI_GET_DESCRIPTOR_TYPE (Node) != ACPI_DESC_TYPE_NAMED)
 692         {
 693             AcpiOsPrintf ("Invalid Descriptor Type for Node %p [%s] - is %2.2X s
 694                 Node, AcpiUtGetDescriptorName (Node), ACPI_GET_DESCRIPTOR_TYPE (
 695                 ACPI_DESC_TYPE_NAMED);
 696             return (AE_OK);
 697         }

 699         if ((Node->Type == ACPI_TYPE_LOCAL_ALIAS)  ||
 700             (Node->Type == ACPI_TYPE_LOCAL_METHOD_ALIAS))
 701         {
 702             Node = (ACPI_NAMESPACE_NODE *) Node->Object;
 703         }
 704         else
 705         {
 706             Alias = FALSE;
 707         }
 708     }

 710     if (Node->Type > ACPI_TYPE_LOCAL_MAX)
 711     {
 712         AcpiOsPrintf ("Invalid Object Type for Node %p, Type = %X\n",
 713             Node, Node->Type);
 714         return (AE_OK);
 715     }

 717     if (!AcpiUtValidAcpiName (Node->Name.Ascii))
 666     if (!AcpiUtValidAcpiName (Node->Name.Integer))
 718     {
 719         AcpiOsPrintf ("Invalid AcpiName for Node %p\n", Node);
 720         return (AE_OK);
 721     }

 723     Object = AcpiNsGetAttachedObject (Node);
 724     if (Object)
 725     {
 726         Info->Objects++;
 727         if (ACPI_GET_DESCRIPTOR_TYPE (Object) != ACPI_DESC_TYPE_OPERAND)
 728         {
 729             AcpiOsPrintf ("Invalid Descriptor Type for Object %p [%s]\n",
 730                 Object, AcpiUtGetDescriptorName (Object));
 731         }
 732     }

 734     return (AE_OK);
```



```
 735 }
______unchanged_portion_omitted_


 812 /*****************************************************************************
 813  *
 814  * FUNCTION:    AcpiDbFindReferences
 815  *
 816  * PARAMETERS:  ObjectArg       - String with hex value of the object
 817  *
 818  * RETURN:      None
 819  *
 820  * DESCRIPTION: Search namespace for all references to the input object
 821  *
 822  ****************************************************************************/

 824 void
 825 AcpiDbFindReferences (
 826     char                    *ObjectArg)
 827 {
 828     ACPI_OPERAND_OBJECT     *ObjDesc;
 829     ACPI_SIZE               Address;


 832     /* Convert string to object pointer */

 834     Address = ACPI_STRTOUL (ObjectArg, NULL, 16);
 835     ObjDesc = ACPI_TO_POINTER (Address);
 782     ObjDesc = ACPI_TO_POINTER (ACPI_STRTOUL (ObjectArg, NULL, 16));

 837     /* Search all nodes in namespace */

 839     (void) AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT, ACPI_UINT32_MAX,
 840                     AcpiDbWalkForReferences, NULL, (void *) ObjDesc, NULL);
 841 }
______unchanged_portion_omitted_
```

```
**********************************************************
   17059 Thu Dec 26 13:48:44 2013
new/usr/src/common/acpica/components/debugger/dbstats.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dbstats - Generation and display of ACPI table statistics
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #include "acpi.h"
  46 #include "accommon.h"
  47 #include "acdebug.h"
  48 #include "acnamesp.h"

  50 #ifdef ACPI_DEBUGGER

  52 #define _COMPONENT          ACPI_CA_DEBUGGER
  53         ACPI_MODULE_NAME    ("dbstats")

  55 /* Local prototypes */

  57 static void
  58 AcpiDbCountNamespaceObjects (
  59     void);
```



```
  61 static void
  62 AcpiDbEnumerateObject (
  63     ACPI_OPERAND_OBJECT     *ObjDesc);

  65 static ACPI_STATUS
  66 AcpiDbClassifyOneObject (
  67     ACPI_HANDLE             ObjHandle,
  68     UINT32                  NestingLevel,
  69     void                    *Context,
  70     void                    **ReturnValue);

  72 #if defined ACPI_DBG_TRACK_ALLOCATIONS || defined ACPI_USE_LOCAL_CACHE
  73 static void
  74 AcpiDbListInfo (
  75     ACPI_MEMORY_LIST        *List);
  76 #endif


  79 /*
  80  * Statistics subcommands
  81  */
  82 static ACPI_DB_ARGUMENT_INFO    AcpiDbStatTypes [] =
  82 static ARGUMENT_INFO        AcpiDbStatTypes [] =
  83 {
  84     {"ALLOCATIONS"},
  85     {"OBJECTS"},
  86     {"MEMORY"},
  87     {"MISC"},
  88     {"TABLES"},
  89     {"SIZES"},
  90     {"STACK"},
  91     {NULL}           /* Must be null terminated */
  92 };
_____unchanged_portion_omitted_
 174 #endif


 177 /******************************************************************************
 178  *
 179  * FUNCTION:    AcpiDbEnumerateObject
 180  *
 181  * PARAMETERS:  ObjDesc             - Object to be counted
 182  *
 183  * RETURN:      None
 184  *
 185  * DESCRIPTION: Add this object to the global counts, by object type.
 186  *              Limited recursion handles subobjects and packages, and this
 187  *              is probably acceptable within the AML debugger only.
 188  *
 189  *****************************************************************************/

 191 static void
 192 AcpiDbEnumerateObject (
 193     ACPI_OPERAND_OBJECT     *ObjDesc)
 194 {
 195     UINT32                  i;


 198     if (!ObjDesc)
 199     {
 200         return;
 201     }

 203     /* Enumerate this object first */
```

```
205     AcpiGbl_NumObjects++;

207     if (ObjDesc->Common.Type > ACPI_TYPE_NS_NODE_MAX)
208     {
209         AcpiGbl_ObjTypeCountMisc++;
210     }
211     else
212     {
213         AcpiGbl_ObjTypeCount [ObjDesc->Common.Type]++;
214     }

216     /* Count the sub-objects */

218     switch (ObjDesc->Common.Type)
219     {
220     case ACPI_TYPE_PACKAGE:

222         for (i = 0; i < ObjDesc->Package.Count; i++)
223         {
224             AcpiDbEnumerateObject (ObjDesc->Package.Elements[i]);
225         }
226         break;

228     case ACPI_TYPE_DEVICE:

230         AcpiDbEnumerateObject (ObjDesc->Device.NotifyList[0]);
231         AcpiDbEnumerateObject (ObjDesc->Device.NotifyList[1]);
230         AcpiDbEnumerateObject (ObjDesc->Device.SystemNotify);
231         AcpiDbEnumerateObject (ObjDesc->Device.DeviceNotify);
232         AcpiDbEnumerateObject (ObjDesc->Device.Handler);
233         break;

235     case ACPI_TYPE_BUFFER_FIELD:

237         if (AcpiNsGetSecondaryObject (ObjDesc))
238         {
239             AcpiGbl_ObjTypeCount [ACPI_TYPE_BUFFER_FIELD]++;
240         }
241         break;

243     case ACPI_TYPE_REGION:

245         AcpiGbl_ObjTypeCount [ACPI_TYPE_LOCAL_REGION_FIELD ]++;
246         AcpiDbEnumerateObject (ObjDesc->Region.Handler);
247         break;

249     case ACPI_TYPE_POWER:

251         AcpiDbEnumerateObject (ObjDesc->PowerResource.NotifyList[0]);
252         AcpiDbEnumerateObject (ObjDesc->PowerResource.NotifyList[1]);
251         AcpiDbEnumerateObject (ObjDesc->PowerResource.SystemNotify);
252         AcpiDbEnumerateObject (ObjDesc->PowerResource.DeviceNotify);
253         break;

255     case ACPI_TYPE_PROCESSOR:

257         AcpiDbEnumerateObject (ObjDesc->Processor.NotifyList[0]);
258         AcpiDbEnumerateObject (ObjDesc->Processor.NotifyList[1]);
257         AcpiDbEnumerateObject (ObjDesc->Processor.SystemNotify);
258         AcpiDbEnumerateObject (ObjDesc->Processor.DeviceNotify);
259         AcpiDbEnumerateObject (ObjDesc->Processor.Handler);
260         break;

262     case ACPI_TYPE_THERMAL:

264         AcpiDbEnumerateObject (ObjDesc->ThermalZone.NotifyList[0]);
```



```
265         AcpiDbEnumerateObject (ObjDesc->ThermalZone.NotifyList[1]);
264         AcpiDbEnumerateObject (ObjDesc->ThermalZone.SystemNotify);
265         AcpiDbEnumerateObject (ObjDesc->ThermalZone.DeviceNotify);
266         AcpiDbEnumerateObject (ObjDesc->ThermalZone.Handler);
267         break;

269     default:

271         break;
272     }
273 }


276 /*******************************************************************************
277  *
278  * FUNCTION:    AcpiDbClassifyOneObject
279  *
280  * PARAMETERS:  Callback for WalkNamespace
281  *
282  * RETURN:      Status
283  *
284  * DESCRIPTION: Enumerate both the object descriptor (including subobjects) and
285  *              the parent namespace node.
286  *
287  ******************************************************************************/

289 static ACPI_STATUS
290 AcpiDbClassifyOneObject (
291     ACPI_HANDLE             ObjHandle,
292     UINT32                  NestingLevel,
293     void                    *Context,
294     void                    **ReturnValue)
295 {
296     ACPI_NAMESPACE_NODE     *Node;
297     ACPI_OPERAND_OBJECT     *ObjDesc;
298     UINT32                  Type;


301     AcpiGbl_NumNodes++;

303     Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
304     ObjDesc = AcpiNsGetAttachedObject (Node);

306     AcpiDbEnumerateObject (ObjDesc);

308     Type = Node->Type;
309     if (Type > ACPI_TYPE_NS_NODE_MAX)
310     {
311         AcpiGbl_NodeTypeCountMisc++;
312     }
313     else
314     {
315         AcpiGbl_NodeTypeCount [Type]++;
316     }

318     return (AE_OK);
317     return AE_OK;


321 #ifdef ACPI_FUTURE_IMPLEMENTATION

323     /* TBD: These need to be counted during the initial parsing phase */

325     if (AcpiPsIsNamedOp (Op->Opcode))
326     {
327         NumNodes++;
```

```
328     }

330     if (IsMethod)
331     {
332         NumMethodElements++;
333     }

335     NumGrammarElements++;
336     Op = AcpiPsGetDepthNext (Root, Op);

338     SizeOfParseTree   = (NumGrammarElements - NumMethodElements) *
339                         (UINT32) sizeof (ACPI_PARSE_OBJECT);
340     SizeOfMethodTrees = NumMethodElements * (UINT32) sizeof (ACPI_PARSE_OBJECT);
341     SizeOfNodeEntries = NumNodes * (UINT32) sizeof (ACPI_NAMESPACE_NODE);
342     SizeOfAcpiObjects = NumNodes * (UINT32) sizeof (ACPI_OPERAND_OBJECT);
343 #endif
344 }
```

unchanged_portion_omitted_

```
382 /*****************************************************************************
383  *
384  * FUNCTION:    AcpiDbDisplayStatistics
385  *
386  * PARAMETERS:  TypeArg         - Subcommand
387  *
388  * RETURN:      Status
389  *
390  * DESCRIPTION: Display various statistics
391  *
392  ****************************************************************************/

394 ACPI_STATUS
395 AcpiDbDisplayStatistics (
396     char                    *TypeArg)
397 {
398     UINT32                  i;
399     UINT32                  Temp;


401     if (!TypeArg)
402     {
403         AcpiOsPrintf ("The following subcommands are available:\n    ALLOCATIONS
404         return (AE_OK);
405     }

402     AcpiUtStrupr (TypeArg);
403     Temp = AcpiDbMatchArgument (TypeArg, AcpiDbStatTypes);
404     if (Temp == (UINT32) -1)
405     {
406         AcpiOsPrintf ("Invalid or unsupported argument\n");
407         return (AE_OK);
408     }


411     switch (Temp)
412     {
413     case CMD_STAT_ALLOCATIONS:

415 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
416         AcpiUtDumpAllocationInfo ();
417 #endif
418         break;

420     case CMD_STAT_TABLES:
```



```
422         AcpiOsPrintf ("ACPI Table Information (not implemented):\n\n");
423         break;

425     case CMD_STAT_OBJECTS:

427         AcpiDbCountNamespaceObjects ();

429         AcpiOsPrintf ("\nObjects defined in the current namespace:\n\n");

431         AcpiOsPrintf ("%16.16s %10.10s %10.10s\n",
432             "ACPI_TYPE", "NODES", "OBJECTS");

434         for (i = 0; i < ACPI_TYPE_NS_NODE_MAX; i++)
435         {
436             AcpiOsPrintf ("%16.16s % 10ld% 10ld\n", AcpiUtGetTypeName (i),
437                 AcpiGbl_NodeTypeCount [i], AcpiGbl_ObjTypeCount [i]);
438         }
439         AcpiOsPrintf ("%16.16s % 10ld% 10ld\n", "Misc/Unknown",
440             AcpiGbl_NodeTypeCountMisc, AcpiGbl_ObjTypeCountMisc);

442         AcpiOsPrintf ("%16.16s % 10ld% 10ld\n", "TOTALS:",
443             AcpiGbl_NumNodes, AcpiGbl_NumObjects);
444         break;

446     case CMD_STAT_MEMORY:

448 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
449         AcpiOsPrintf ("\n----Object Statistics (all in hex)---------\n");

451         AcpiDbListInfo (AcpiGbl_GlobalList);
452         AcpiDbListInfo (AcpiGbl_NsNodeList);
453 #endif

455 #ifdef ACPI_USE_LOCAL_CACHE
456         AcpiOsPrintf ("\n----Cache Statistics (all in hex)---------\n");
457         AcpiDbListInfo (AcpiGbl_OperandCache);
458         AcpiDbListInfo (AcpiGbl_PsNodeCache);
459         AcpiDbListInfo (AcpiGbl_PsNodeExtCache);
460         AcpiDbListInfo (AcpiGbl_StateCache);
461 #endif

463         break;

465     case CMD_STAT_MISC:

467         AcpiOsPrintf ("\nMiscellaneous Statistics:\n\n");
468         AcpiOsPrintf ("Calls to AcpiPsFind:..  ........% 7ld\n",
469             AcpiGbl_PsFindCount);
470         AcpiOsPrintf ("Calls to AcpiNsLookup:..........% 7ld\n",
471             AcpiGbl_NsLookupCount);

473         AcpiOsPrintf ("\n");

475         AcpiOsPrintf ("Mutex usage:\n\n");
476         for (i = 0; i < ACPI_NUM_MUTEX; i++)
477         {
478             AcpiOsPrintf ("%-28s:       % 7ld\n",
479                 AcpiUtGetMutexName (i), AcpiGbl_MutexInfo[i].UseCount);
480         }
481         break;


483     case CMD_STAT_SIZES:

485         AcpiOsPrintf ("\nInternal object sizes:\n\n");
```

```
487         AcpiOsPrintf ("Common           %3d\n", sizeof (ACPI_OBJECT_COMMON));
488         AcpiOsPrintf ("Number           %3d\n", sizeof (ACPI_OBJECT_INTEGER));
489         AcpiOsPrintf ("String           %3d\n", sizeof (ACPI_OBJECT_STRING));
490         AcpiOsPrintf ("Buffer           %3d\n", sizeof (ACPI_OBJECT_BUFFER));
491         AcpiOsPrintf ("Package          %3d\n", sizeof (ACPI_OBJECT_PACKAGE));
492         AcpiOsPrintf ("BufferField      %3d\n", sizeof (ACPI_OBJECT_BUFFER_FIELD
493         AcpiOsPrintf ("Device           %3d\n", sizeof (ACPI_OBJECT_DEVICE));
494         AcpiOsPrintf ("Event            %3d\n", sizeof (ACPI_OBJECT_EVENT));
495         AcpiOsPrintf ("Method           %3d\n", sizeof (ACPI_OBJECT_METHOD));
496         AcpiOsPrintf ("Mutex            %3d\n", sizeof (ACPI_OBJECT_MUTEX));
497         AcpiOsPrintf ("Region           %3d\n", sizeof (ACPI_OBJECT_REGION));
498         AcpiOsPrintf ("PowerResource    %3d\n", sizeof (ACPI_OBJECT_POWER_RESOUR
499         AcpiOsPrintf ("Processor        %3d\n", sizeof (ACPI_OBJECT_PROCESSOR));
500         AcpiOsPrintf ("ThermalZone      %3d\n", sizeof (ACPI_OBJECT_THERMAL_ZONE
501         AcpiOsPrintf ("RegionField      %3d\n", sizeof (ACPI_OBJECT_REGION_FIELD
502         AcpiOsPrintf ("BankField        %3d\n", sizeof (ACPI_OBJECT_BANK_FIELD))
503         AcpiOsPrintf ("IndexField       %3d\n", sizeof (ACPI_OBJECT_INDEX_FIELD)
504         AcpiOsPrintf ("Reference        %3d\n", sizeof (ACPI_OBJECT_REFERENCE));
505         AcpiOsPrintf ("Notify           %3d\n", sizeof (ACPI_OBJECT_NOTIFY_HANDL
506         AcpiOsPrintf ("AddressSpace     %3d\n", sizeof (ACPI_OBJECT_ADDR_HANDLER
507         AcpiOsPrintf ("Extra            %3d\n", sizeof (ACPI_OBJECT_EXTRA));
508         AcpiOsPrintf ("Data             %3d\n", sizeof (ACPI_OBJECT_DATA));

510         AcpiOsPrintf ("\n");

512         AcpiOsPrintf ("ParseObject      %3d\n", sizeof (ACPI_PARSE_OBJ_COMMON));
513         AcpiOsPrintf ("ParseObjectNamed %3d\n", sizeof (ACPI_PARSE_OBJ_NAMED));
514         AcpiOsPrintf ("ParseObjectAsl   %3d\n", sizeof (ACPI_PARSE_OBJ_ASL));
515         AcpiOsPrintf ("OperandObject    %3d\n", sizeof (ACPI_OPERAND_OBJECT));
516         AcpiOsPrintf ("NamespaceNode    %3d\n", sizeof (ACPI_NAMESPACE_NODE));
517         AcpiOsPrintf ("AcpiObject       %3d\n", sizeof (ACPI_OBJECT));

519         AcpiOsPrintf ("\n");

521         AcpiOsPrintf ("Generic State    %3d\n", sizeof (ACPI_GENERIC_STATE));
522         AcpiOsPrintf ("Common State     %3d\n", sizeof (ACPI_COMMON_STATE));
523         AcpiOsPrintf ("Control State    %3d\n", sizeof (ACPI_CONTROL_STATE));
524         AcpiOsPrintf ("Update State     %3d\n", sizeof (ACPI_UPDATE_STATE));
525         AcpiOsPrintf ("Scope State      %3d\n", sizeof (ACPI_SCOPE_STATE));
526         AcpiOsPrintf ("Parse Scope      %3d\n", sizeof (ACPI_PSCOPE_STATE));
527         AcpiOsPrintf ("Package State    %3d\n", sizeof (ACPI_PKG_STATE));
528         AcpiOsPrintf ("Thread State     %3d\n", sizeof (ACPI_THREAD_STATE));
529         AcpiOsPrintf ("Result Values    %3d\n", sizeof (ACPI_RESULT_VALUES));
530         AcpiOsPrintf ("Notify Info      %3d\n", sizeof (ACPI_NOTIFY_INFO));
531         break;


533     case CMD_STAT_STACK:
534 #if defined(ACPI_DEBUG_OUTPUT)

536         Temp = (UINT32) ACPI_PTR_DIFF (AcpiGbl_EntryStackPointer, AcpiGbl_Lowest

538         AcpiOsPrintf ("\nSubsystem Stack Usage:\n\n");
539         AcpiOsPrintf ("Entry Stack Pointer          %p\n", AcpiGbl_EntryStackPoi
540         AcpiOsPrintf ("Lowest Stack Pointer         %p\n", AcpiGbl_LowestStackPo
541         AcpiOsPrintf ("Stack Use                    %X (%u)\n", Temp, Temp);
542         AcpiOsPrintf ("Deepest Procedure Nesting    %u\n", AcpiGbl_DeepestNestin
543 #endif
544         break;

546     default:

548         break;
549     }

551     AcpiOsPrintf ("\n");
```



```
552     return (AE_OK);
553 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   13449 Thu Dec 26 13:48:44 2013
new/usr/src/common/acpica/components/debugger/dbutils.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbutils - AML debugger utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acdebug.h"
#include "acdisasm.h"


#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbutils")

/* Local prototypes */

#ifdef ACPI_OBSOLETE_FUNCTIONS
```



```
ACPI_STATUS
AcpiDbSecondPassParse (
    ACPI_PARSE_OBJECT       *Root);

void
AcpiDbDumpBuffer (
    UINT32                  Address);
#endif

static char                 *Converter = "0123456789ABCDEF";


/******************************************************************************
 *
 * FUNCTION:    AcpiDbMatchArgument
 *
 * PARAMETERS:  UserArgument            - User command line
 *              Arguments               - Array of commands to match against
 *
 * RETURN:      Index into command array or ACPI_TYPE_NOT_FOUND if not found
 *
 * DESCRIPTION: Search command array for a command match
 *
 *****************************************************************************/

ACPI_OBJECT_TYPE
AcpiDbMatchArgument (
    char                    *UserArgument,
    ACPI_DB_ARGUMENT_INFO   *Arguments)
    ARGUMENT_INFO           *Arguments)
{
    UINT32                  i;


    if (!UserArgument || UserArgument[0] == 0)
    {
        return (ACPI_TYPE_NOT_FOUND);
    }

    for (i = 0; Arguments[i].Name; i++)
    {
        if (ACPI_STRSTR (Arguments[i].Name, UserArgument) == Arguments[i].Name)
        {
            return (i);
        }
    }

    /* Argument not recognized */

    return (ACPI_TYPE_NOT_FOUND);
}
```

_____unchanged_portion_omitted_

```


/******************************************************************************
 *
 * FUNCTION:    AcpiDbDumpExternalObject
 *
 * PARAMETERS:  ObjDesc         - External ACPI object to dump
 *              Level           - Nesting level.
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the contents of an ACPI external object
 *
 *****************************************************************************/
```

```
156 void
157 AcpiDbDumpExternalObject (
158     ACPI_OBJECT             *ObjDesc,
159     UINT32                  Level)
160 {
161     UINT32                  i;


164     if (!ObjDesc)
165     {
166         AcpiOsPrintf ("[Null Object]\n");
167         return;
168     }

170     for (i = 0; i < Level; i++)
171     {
172         AcpiOsPrintf ("  ");
173     }

175     switch (ObjDesc->Type)
176     {
177     case ACPI_TYPE_ANY:

179         AcpiOsPrintf ("[Null Object] (Type=0)\n");
180         break;


182     case ACPI_TYPE_INTEGER:

184         AcpiOsPrintf ("[Integer] = %8.8X%8.8X\n",
185                     ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
186         break;


188     case ACPI_TYPE_STRING:

190         AcpiOsPrintf ("[String] Length %.2X = ", ObjDesc->String.Length);
191         AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
193         for (i = 0; i < ObjDesc->String.Length; i++)
194         {
195             AcpiOsPrintf ("%c", ObjDesc->String.Pointer[i]);
196         }
192         AcpiOsPrintf ("\n");
193         break;


195     case ACPI_TYPE_BUFFER:

197         AcpiOsPrintf ("[Buffer] Length %.2X = ", ObjDesc->Buffer.Length);
198         if (ObjDesc->Buffer.Length)
199         {
200             if (ObjDesc->Buffer.Length > 16)
201             {
202                 AcpiOsPrintf ("\n");
203             }
204             AcpiUtDebugDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer
205                     ObjDesc->Buffer.Length, DB_BYTE_DISPLAY, _COMPONENT);
210             AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
211                     ObjDesc->Buffer.Length, DB_DWORD_DISPLAY, _COMPONENT);
206         }
207         else
208         {
209             AcpiOsPrintf ("\n");
210         }
211         break;
```



```
213     case ACPI_TYPE_PACKAGE:

215         AcpiOsPrintf ("[Package] Contains %u Elements:\n",
216                 ObjDesc->Package.Count);

218         for (i = 0; i < ObjDesc->Package.Count; i++)
219         {
220             AcpiDbDumpExternalObject (&ObjDesc->Package.Elements[i], Level+1);
221         }
222         break;


224     case ACPI_TYPE_LOCAL_REFERENCE:

226         AcpiOsPrintf ("[Object Reference] = ");
227         AcpiDmDisplayInternalObject (ObjDesc->Reference.Handle, NULL);
228         break;


230     case ACPI_TYPE_PROCESSOR:

232         AcpiOsPrintf ("[Processor]\n");
233         break;


235     case ACPI_TYPE_POWER:

237         AcpiOsPrintf ("[Power Resource]\n");
238         break;


240     default:

242         AcpiOsPrintf ("[Unknown Type] %X\n", ObjDesc->Type);
243         break;
244     }
245 }


248 /*******************************************************************************
249  *
250  * FUNCTION:    AcpiDbPrepNamestring
251  *
252  * PARAMETERS:  Name            - String to prepare
253  *
254  * RETURN:      None
255  *
256  * DESCRIPTION: Translate all forward slashes and dots to backslashes.
257  *
258  ******************************************************************************/

260 void
261 AcpiDbPrepNamestring (
262     char                    *Name)
263 {

265     if (!Name)
266     {
267         return;
268     }

270     AcpiUtStrupr (Name);

272     /* Convert a leading forward slash to a backslash */
```

```
274     if (*Name == '/')
275     {
276         *Name = '\\';
277     }

279     /* Ignore a leading backslash, this is the root prefix */

281     if (ACPI_IS_ROOT_PREFIX (*Name))
292     if (*Name == '\\')
282     {
283         Name++;
284     }

286     /* Convert all slash path separators to dots */

288     while (*Name)
289     {
290         if ((*Name == '/') ||
291             (*Name == '\\'))
292         {
293             *Name = '.';
294         }

296         Name++;
297     }
298 }
```

*_____unchanged_portion_omitted_*

```
353 /*******************************************************************************
354  *
355  * FUNCTION:    AcpiDbUint32ToHexString
366  * FUNCTION:    AcpiDbUInt32ToHexString
356  *
357  * PARAMETERS:  Value           - The value to be converted to string
358  *              Buffer          - Buffer for result (not less than 11 bytes)
359  *
360  * RETURN:      None
361  *
362  * DESCRIPTION: Convert the unsigned 32-bit value to the hexadecimal image
363  *
364  * NOTE: It is the caller's responsibility to ensure that the length of buffer
365  *       is sufficient.
366  *
367  ******************************************************************************/

369 void
370 AcpiDbUint32ToHexString (
381 AcpiDbUInt32ToHexString (
371     UINT32                  Value,
372     char                    *Buffer)
373 {
374     int                     i;


377     if (Value == 0)
378     {
379         ACPI_STRCPY (Buffer, "0");
380         return;
381     }

383     Buffer[8] = '\0';

385     for (i = 7; i >= 0; i--)
386     {
```



```
387         Buffer[i] = Converter [Value & 0x0F];
388         Value = Value >> 4;
389     }
390 }
```

*_____unchanged_portion_omitted_*

```
488 /*******************************************************************************
489  *
490  * FUNCTION:    AcpiDbDumpBuffer
491  *
492  * PARAMETERS:  Address             - Pointer to the buffer
493  *
494  * RETURN:      None
495  *
496  * DESCRIPTION: Print a portion of a buffer
497  *
498  ******************************************************************************/

500 void
501 AcpiDbDumpBuffer (
502     UINT32                  Address)
503 {

505     AcpiOsPrintf ("\nLocation %X:\n", Address);

507     AcpiDbgLevel |= ACPI_LV_TABLES;
508     AcpiUtDebugDumpBuffer (ACPI_TO_POINTER (Address), 64, DB_BYTE_DISPLAY,
519     AcpiUtDumpBuffer (ACPI_TO_POINTER (Address), 64, DB_BYTE_DISPLAY,
509             ACPI_UINT32_MAX);
510 }
511 #endif

513 #endif /* ACPI_DEBUGGER */
```

```
**********************************************************
   15621 Thu Dec 26 13:48:44 2013
new/usr/src/common/acpica/components/debugger/dbxface.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dbxface - AML Debugger external interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acdebug.h"
#include "acdisasm.h"


#ifdef ACPI_DEBUGGER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbxface")


/* Local prototypes */
```



```
static ACPI_STATUS
AcpiDbStartCommand (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

#ifdef ACPI_OBSOLETE_FUNCTIONS
void
AcpiDbMethodEnd (
    ACPI_WALK_STATE         *WalkState);
#endif


/******************************************************************************
 *
 * FUNCTION:    AcpiDbStartCommand
 *
 * PARAMETERS:  WalkState       - Current walk
 *              Op              - Current executing Op, from AML interpreter
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Enter debugger command loop
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiDbStartCommand (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_STATUS             Status;


    /* TBD: [Investigate] are there namespace locking issues here? */

    /* AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE); */

    /* Go into the command loop and await next user command */


    AcpiGbl_MethodExecuting = TRUE;
    Status = AE_CTRL_TRUE;
    while (Status == AE_CTRL_TRUE)
    {
        if (AcpiGbl_DebuggerConfiguration == DEBUGGER_MULTI_THREADED)
        {
            /* Handshake with the front-end that gets user command lines */

            Status = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
            Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_READY);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
        }
        else
        {
            /* Single threaded, we must get a command line ourselves */

            /* Force output to console until a command is entered */
```

```
125             AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);

127             /* Different prompt if method is executing */

129             if (!AcpiGbl_MethodExecuting)
130             {
131                 AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_COMMAND_PROMPT);
132             }
133             else
134             {
135                 AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_EXECUTE_PROMPT);
136             }

138             /* Get the user input line */

140             Status = AcpiOsGetLine (AcpiGbl_DbLineBuf,
141                 ACPI_DB_LINE_BUFFER_SIZE, NULL);
142             if (ACPI_FAILURE (Status))
143             {
144                 ACPI_EXCEPTION ((AE_INFO, Status, "While parsing command line"));
145                 return (Status);
146             }
147         }

149         Status = AcpiDbCommandDispatch (AcpiGbl_DbLineBuf, WalkState, Op);
150     }

152     /* AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE); */

154     return (Status);
155 }


158 /*******************************************************************************
159  *
160  * FUNCTION:    AcpiDbSingleStep
161  *
162  * PARAMETERS:  WalkState       - Current walk
163  *              Op              - Current executing op (from aml interpreter)
164  *              OpcodeClass     - Class of the current AML Opcode
165  *
166  * RETURN:      Status
167  *
168  * DESCRIPTION: Called just before execution of an AML opcode.
169  *
170  ******************************************************************************/

172 ACPI_STATUS
173 AcpiDbSingleStep (
174     ACPI_WALK_STATE         *WalkState,
175     ACPI_PARSE_OBJECT       *Op,
176     UINT32                  OpcodeClass)
177 {
178     ACPI_PARSE_OBJECT       *Next;
179     ACPI_STATUS             Status = AE_OK;
180     UINT32                  OriginalDebugLevel;
181     ACPI_PARSE_OBJECT       *DisplayOp;
182     ACPI_PARSE_OBJECT       *ParentOp;


185     ACPI_FUNCTION_ENTRY ();


188     /* Check the abort flag */

190     if (AcpiGbl_AbortMethod)
```



```
191     {
192         AcpiGbl_AbortMethod = FALSE;
193         return (AE_ABORT_METHOD);
194     }

196     /* Check for single-step breakpoint */

198     if (WalkState->MethodBreakpoint &&
199        (WalkState->MethodBreakpoint <= Op->Common.AmlOffset))
200     {
201         /* Check if the breakpoint has been reached or passed */
202         /* Hit the breakpoint, resume single step, reset breakpoint */

204         AcpiOsPrintf ("***Break*** at AML offset %X\n", Op->Common.AmlOffset);
205         AcpiGbl_CmSingleStep = TRUE;
206         AcpiGbl_StepToNextCall = FALSE;
207         WalkState->MethodBreakpoint = 0;
208     }

210     /* Check for user breakpoint (Must be on exact Aml offset) */

212     else if (WalkState->UserBreakpoint &&
213             (WalkState->UserBreakpoint == Op->Common.AmlOffset))
214     {
215         AcpiOsPrintf ("***UserBreakpoint*** at AML offset %X\n",
216             Op->Common.AmlOffset);
217         AcpiGbl_CmSingleStep = TRUE;
218         AcpiGbl_StepToNextCall = FALSE;
219         WalkState->MethodBreakpoint = 0;
220     }

222     /*
223      * Check if this is an opcode that we are interested in --
224      * namely, opcodes that have arguments
225      */
226     if (Op->Common.AmlOpcode == AML_INT_NAMEDFIELD_OP)
227     {
228         return (AE_OK);
229     }

231     switch (OpcodeClass)
232     {
233     case AML_CLASS_UNKNOWN:
234     case AML_CLASS_ARGUMENT:    /* constants, literals, etc. do nothing */

236         return (AE_OK);

238     default:

240         /* All other opcodes -- continue */
241         break;
242     }

244     /*
245      * Under certain debug conditions, display this opcode and its operands
246      */
247     if ((AcpiGbl_DbOutputToFile)            ||
248         (AcpiGbl_CmSingleStep)              ||
249         (AcpiDbgLevel & ACPI_LV_PARSE))
250     {
251         if ((AcpiGbl_DbOutputToFile)        ||
252             (AcpiDbgLevel & ACPI_LV_PARSE))
253         {
254             AcpiOsPrintf ("\n[AmlDebug] Next AML Opcode to execute:\n");
255         }
```

```
257         /*
258          * Display this op (and only this op - zero out the NEXT field
259          * temporarily, and disable parser trace output for the duration of
260          * the display because we don't want the extraneous debug output)
261          */
262         OriginalDebugLevel = AcpiDbgLevel;
263         AcpiDbgLevel &= ~(ACPI_LV_PARSE | ACPI_LV_FUNCTIONS);
264         Next = Op->Common.Next;
265         Op->Common.Next = NULL;


268         DisplayOp = Op;
269         ParentOp = Op->Common.Parent;
270         if (ParentOp)
271         {
272             if ((WalkState->ControlState) &&
273                 (WalkState->ControlState->Common.State ==
274                     ACPI_CONTROL_PREDICATE_EXECUTING))
275             {
276                 /*
277                  * We are executing the predicate of an IF or WHILE statement
278                  * Search upwards for the containing IF or WHILE so that the
279                  * entire predicate can be displayed.
280                  */
281                 while (ParentOp)
282                 {
283                     if ((ParentOp->Common.AmlOpcode == AML_IF_OP) ||
284                         (ParentOp->Common.AmlOpcode == AML_WHILE_OP))
285                     {
286                         DisplayOp = ParentOp;
287                         break;
288                     }
289                     ParentOp = ParentOp->Common.Parent;
290                 }
291             }
292             else
293             {
294                 while (ParentOp)
295                 {
296                     if ((ParentOp->Common.AmlOpcode == AML_IF_OP)     ||
297                         (ParentOp->Common.AmlOpcode == AML_ELSE_OP)   ||
298                         (ParentOp->Common.AmlOpcode == AML_SCOPE_OP)  ||
299                         (ParentOp->Common.AmlOpcode == AML_METHOD_OP) ||
300                         (ParentOp->Common.AmlOpcode == AML_WHILE_OP))
301                     {
302                         break;
303                     }
304                     DisplayOp = ParentOp;
305                     ParentOp = ParentOp->Common.Parent;
306                 }
307             }
308         }

310         /* Now we can display it */

312         AcpiDmDisassemble (WalkState, DisplayOp, ACPI_UINT32_MAX);

314         if ((Op->Common.AmlOpcode == AML_IF_OP) ||
315             (Op->Common.AmlOpcode == AML_WHILE_OP))
316         {
317             if (WalkState->ControlState->Common.Value)
318             {
319                 AcpiOsPrintf ("Predicate = [True], IF block was executed\n");
320             }
321             else
322             {
```



```
323                 AcpiOsPrintf ("Predicate = [False], Skipping IF block\n");
324             }
325         }
326         else if (Op->Common.AmlOpcode == AML_ELSE_OP)
327         {
328             AcpiOsPrintf ("Predicate = [False], ELSE block was executed\n");
329         }

331         /* Restore everything */

333         Op->Common.Next = Next;
334         AcpiOsPrintf ("\n");
335         if ((AcpiGbl_DbOutputToFile)        ||
336             (AcpiDbgLevel & ACPI_LV_PARSE))
337         {
338             AcpiOsPrintf ("\n");
339         }
340         AcpiDbgLevel = OriginalDebugLevel;
341     }

343     /* If we are not single stepping, just continue executing the method */

345     if (!AcpiGbl_CmSingleStep)
346     {
347         return (AE_OK);
348     }

350     /*
351      * If we are executing a step-to-call command,
352      * Check if this is a method call.
353      */
354     if (AcpiGbl_StepToNextCall)
355     {
356         if (Op->Common.AmlOpcode != AML_INT_METHODCALL_OP)
357         {
358             /* Not a method call, just keep executing */

360             return (AE_OK);
361         }

363         /* Found a method call, stop executing */

365         AcpiGbl_StepToNextCall = FALSE;
366     }

368     /*
369      * If the next opcode is a method call, we will "step over" it
370      * by default.
371      */
372     if (Op->Common.AmlOpcode == AML_INT_METHODCALL_OP)
373     {
374         /* Force no more single stepping while executing called method */

376         AcpiGbl_CmSingleStep = FALSE;

378         /*
379          * Set the breakpoint on/before the call, it will stop execution
380          * as soon as we return
381          */
382         WalkState->MethodBreakpoint = 1;  /* Must be non-zero! */
383     }


386     Status = AcpiDbStartCommand (WalkState, Op);

388     /* User commands complete, continue execution of the interrupted method */
```

```
390     return (Status);
391 }


394 /******************************************************************************
395  *
396  * FUNCTION:    AcpiDbInitialize
397  *
398  * PARAMETERS:  None
399  *
400  * RETURN:      Status
401  *
402  * DESCRIPTION: Init and start debugger
403  *
404  *****************************************************************************/

406 ACPI_STATUS
407 AcpiDbInitialize (
408     void)
409 {
410     ACPI_STATUS             Status;


413     ACPI_FUNCTION_TRACE (DbInitialize);


416     /* Init globals */

418     AcpiGbl_DbBuffer            = NULL;
419     AcpiGbl_DbFilename          = NULL;
420     AcpiGbl_DbOutputToFile      = FALSE;

422     AcpiGbl_DbDebugLevel        = ACPI_LV_VERBOSITY2;
423     AcpiGbl_DbConsoleDebugLevel = ACPI_NORMAL_DEFAULT | ACPI_LV_TABLES;
424     AcpiGbl_DbOutputFlags       = ACPI_DB_CONSOLE_OUTPUT;

426     AcpiGbl_DbOpt_tables        = FALSE;
427     AcpiGbl_DbOpt_disasm        = FALSE;
428     AcpiGbl_DbOpt_stats         = FALSE;
429     AcpiGbl_DbOpt_verbose       = TRUE;
430     AcpiGbl_DbOpt_ini_methods   = TRUE;

432     AcpiGbl_DbBuffer = AcpiOsAllocate (ACPI_DEBUG_BUFFER_SIZE);
433     if (!AcpiGbl_DbBuffer)
434     {
435         return_ACPI_STATUS (AE_NO_MEMORY);
430         return (AE_NO_MEMORY);
436     }
437     ACPI_MEMSET (AcpiGbl_DbBuffer, 0, ACPI_DEBUG_BUFFER_SIZE);

439     /* Initial scope is the root */

441     AcpiGbl_DbScopeBuf [0] = AML_ROOT_PREFIX;
436     AcpiGbl_DbScopeBuf [0] = '\\';
442     AcpiGbl_DbScopeBuf [1] =  0;
443     AcpiGbl_DbScopeNode = AcpiGbl_RootNode;

445     /*
446      * If configured for multi-thread support, the debug executor runs in
447      * a separate thread so that the front end can be in another address
448      * space, environment, or even another machine.
449      */
450     if (AcpiGbl_DebuggerConfiguration & DEBUGGER_MULTI_THREADED)
451     {
452         /* These were created with one unit, grab it */
```



```
454         Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
455         if (ACPI_FAILURE (Status))
456         {
457             AcpiOsPrintf ("Could not get debugger mutex\n");
458             return_ACPI_STATUS (Status);
453             return (Status);
459         }

461         Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_READY);
462         if (ACPI_FAILURE (Status))
463         {
464             AcpiOsPrintf ("Could not get debugger mutex\n");
465             return_ACPI_STATUS (Status);
460             return (Status);
466         }

468         /* Create the debug execution thread to execute commands */

470         Status = AcpiOsExecute (OSL_DEBUGGER_THREAD, AcpiDbExecuteThread, NULL);
471         if (ACPI_FAILURE (Status))
472         {
473             ACPI_EXCEPTION ((AE_INFO, Status, "Could not start debugger thread")
474             return_ACPI_STATUS (Status);
468             AcpiOsPrintf ("Could not start debugger thread\n");
469             return (Status);
475         }
476     }

478     if (!AcpiGbl_DbOpt_verbose)
479     {
480         AcpiGbl_DbOpt_disasm = TRUE;
481         AcpiGbl_DbOpt_stats = FALSE;
482     }

484     return_ACPI_STATUS (AE_OK);
479     return (AE_OK);
485 }


488 /******************************************************************************
489  *
490  * FUNCTION:    AcpiDbTerminate
491  *
492  * PARAMETERS:  None
493  *
494  * RETURN:      None
495  *
496  * DESCRIPTION: Stop debugger
497  *
498  *****************************************************************************/

500 void
501 AcpiDbTerminate (
502     void)
503 {

505     if (AcpiGbl_DbBuffer)
506     {
507         AcpiOsFree (AcpiGbl_DbBuffer);
508         AcpiGbl_DbBuffer = NULL;
509     }

511     /* Ensure that debug output is now disabled */

513     AcpiGbl_DbOutputFlags = ACPI_DB_DISABLE_OUTPUT;
```

```
 514 }
______unchanged_portion_omitted_
```

```
**********************************************************
   18848 Thu Dec 26 13:48:45 2013
new/usr/src/common/acpica/components/disassembler/dmbuffer.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmbuffer - AML disassembler, buffer and string support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"
#include "acparser.h"
#include "amlcode.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmbuffer")

/* Local prototypes */

static void
AcpiDmUnicode (
```



```
    ACPI_PARSE_OBJECT       *Op);

static void
AcpiDmIsEisaIdElement (
    ACPI_PARSE_OBJECT       *Op);

static void
AcpiDmPldBuffer (
    UINT32                  Level,
    UINT8                   *ByteData,
    UINT32                  ByteCount);


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDisasmByteList
 *
 * PARAMETERS:  Level               - Current source code indentation level
 *              ByteData            - Pointer to the byte list
 *              ByteCount           - Length of the byte list
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump an AML "ByteList" in Hex format. 8 bytes per line, prefixed
 *              with the hex buffer offset.
 *
 ******************************************************************************/

void
AcpiDmDisasmByteList (
    UINT32                  Level,
    UINT8                   *ByteData,
    UINT32                  ByteCount)
{
    UINT32                  i;


    if (!ByteCount)
    {
        return;
    }

    /* Dump the byte list */

    for (i = 0; i < ByteCount; i++)
    {
        /* New line every 8 bytes */

        if (((i % 8) == 0) && (i < ByteCount))
        {
            if (i > 0)
            {
                AcpiOsPrintf ("\n");
            }

            AcpiDmIndent (Level);
            if (ByteCount > 8)
            {
                AcpiOsPrintf ("/* %04X */  ", i);
            }
        }

        AcpiOsPrintf (" 0x%2.2X", (UINT32) ByteData[i]);

        /* Add comma if there are more bytes to display */
```

```
127         if (i < (ByteCount -1))
128         {
129             AcpiOsPrintf (",");
130         }
131     }
132
133     if (Level)
134     {
135         AcpiOsPrintf ("\n");
136     }
137 }
138
139
140 /*******************************************************************************
141  *
142  * FUNCTION:    AcpiDmByteList
143  *
144  * PARAMETERS:  Info            - Parse tree walk info
145  *              Op              - Byte list op
146  *
147  * RETURN:      None
148  *
149  * DESCRIPTION: Dump a buffer byte list, handling the various types of buffers.
150  *              Buffer type must be already set in the Op DisasmOpcode.
151  *
152  ******************************************************************************/
153
154 void
155 AcpiDmByteList (
156     ACPI_OP_WALK_INFO       *Info,
157     ACPI_PARSE_OBJECT       *Op)
158 {
159     UINT8                   *ByteData;
160     UINT32                  ByteCount;
161
162
163     ByteData = Op->Named.Data;
164     ByteCount = (UINT32) Op->Common.Value.Integer;
165
166     /*
167      * The byte list belongs to a buffer, and can be produced by either
168      * a ResourceTemplate, Unicode, quoted string, or a plain byte list.
169      */
170     switch (Op->Common.Parent->Common.DisasmOpcode)
171     {
172     case ACPI_DASM_RESOURCE:
173
174         AcpiDmResourceTemplate (Info, Op->Common.Parent, ByteData, ByteCount);
175         break;
176
177     case ACPI_DASM_STRING:
178
179         AcpiDmIndent (Info->Level);
180         AcpiUtPrintString ((char *) ByteData, ACPI_UINT16_MAX);
181         AcpiOsPrintf ("\n");
182         break;
183
184     case ACPI_DASM_UNICODE:
185
186         AcpiDmUnicode (Op);
187         break;
188
189     case ACPI_DASM_PLD_METHOD:
190
191         AcpiDmDisasmByteList (Info->Level, ByteData, ByteCount);
192         AcpiDmPldBuffer (Info->Level, ByteData, ByteCount);
```



```
193         break;
194
195     case ACPI_DASM_BUFFER:
196     default:
197         /*
198          * Not a resource, string, or unicode string.
199          * Just dump the buffer
200          */
201         AcpiDmDisasmByteList (Info->Level, ByteData, ByteCount);
202         break;
203     }
204 }
205
206
207 /*******************************************************************************
208  *
209  * FUNCTION:    AcpiDmIsUnicodeBuffer
210  *
211  * PARAMETERS:  Op              - Buffer Object to be examined
212  *
213  * RETURN:      TRUE if buffer contains a UNICODE string
214  *
215  * DESCRIPTION: Determine if a buffer Op contains a Unicode string
216  *
217  ******************************************************************************/
218
219 BOOLEAN
220 AcpiDmIsUnicodeBuffer (
221     ACPI_PARSE_OBJECT       *Op)
222 {
223     UINT8                   *ByteData;
224     UINT32                  ByteCount;
225     UINT32                  WordCount;
226     ACPI_PARSE_OBJECT       *SizeOp;
227     ACPI_PARSE_OBJECT       *NextOp;
228     UINT32                  i;
229
230
231     /* Buffer size is the buffer argument */
232
233     SizeOp = Op->Common.Value.Arg;
234
235     /* Next, the initializer byte list to examine */
236
237     NextOp = SizeOp->Common.Next;
238     if (!NextOp)
239     {
240         return (FALSE);
241     }
242
243     /* Extract the byte list info */
244
245     ByteData = NextOp->Named.Data;
246     ByteCount = (UINT32) NextOp->Common.Value.Integer;
247     WordCount = ACPI_DIV_2 (ByteCount);
248
249     /*
250      * Unicode string must have an even number of bytes and last
251      * word must be zero
252      */
253     if ((!ByteCount)     ||
254         (ByteCount < 4) ||
255         (ByteCount & 1) ||
256         ((UINT16 *) (void *) ByteData)[WordCount - 1] != 0)
257     {
258         return (FALSE);
```

```
    }

    /* For each word, 1st byte must be ascii, 2nd byte must be zero */

    for (i = 0; i < (ByteCount - 2); i += 2)
    {
        if ((!ACPI_IS_PRINT (ByteData[i])) ||
            (ByteData[(ACPI_SIZE) i + 1] != 0))
        {
            return (FALSE);
        }
    }

    /* Ignore the Size argument in the disassembly of this buffer op */

    SizeOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
    return (TRUE);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmIsStringBuffer
 *
 * PARAMETERS:  Op              - Buffer Object to be examined
 *
 * RETURN:      TRUE if buffer contains a ASCII string, FALSE otherwise
 *
 * DESCRIPTION: Determine if a buffer Op contains a ASCII string
 *
 ******************************************************************************/

BOOLEAN
AcpiDmIsStringBuffer (
    ACPI_PARSE_OBJECT       *Op)
{
    UINT8                   *ByteData;
    UINT32                  ByteCount;
    ACPI_PARSE_OBJECT       *SizeOp;
    ACPI_PARSE_OBJECT       *NextOp;
    UINT32                  i;


    /* Buffer size is the buffer argument */

    SizeOp = Op->Common.Value.Arg;

    /* Next, the initializer byte list to examine */

    NextOp = SizeOp->Common.Next;
    if (!NextOp)
    {
        return (FALSE);
    }

    /* Extract the byte list info */

    ByteData = NextOp->Named.Data;
    ByteCount = (UINT32) NextOp->Common.Value.Integer;

    /* Last byte must be the null terminator */

    if ((!ByteCount)     ||
        (ByteCount < 2) ||
        (ByteData[ByteCount-1] != 0))
    {
```



```
        return (FALSE);
    }

    for (i = 0; i < (ByteCount - 1); i++)
    {
        /* TBD: allow some escapes (non-ascii chars).
         * they will be handled in the string output routine
         */

        if (!ACPI_IS_PRINT (ByteData[i]))
        {
            return (FALSE);
        }
    }

    return (TRUE);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmIsPldBuffer
 *
 * PARAMETERS:  Op                  - Buffer Object to be examined
 *
 * RETURN:      TRUE if buffer contains a ASCII string, FALSE otherwise
 *
 * DESCRIPTION: Determine if a buffer Op contains a _PLD structure
 *
 ******************************************************************************/

BOOLEAN
AcpiDmIsPldBuffer (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_PARSE_OBJECT       *ParentOp;


    ParentOp = Op->Common.Parent;
    if (!ParentOp)
    {
        return (FALSE);
    }

    /* Check for form: Name(_PLD, Buffer() {}). Not legal, however */

    if (ParentOp->Common.AmlOpcode == AML_NAME_OP)
    {
        Node = ParentOp->Common.Node;

        if (ACPI_COMPARE_NAME (Node->Name.Ascii, METHOD_NAME__PLD))
        {
            return (TRUE);
        }

        return (FALSE);
    }

    /* Check for proper form: Name(_PLD, Package() {Buffer() {}}) */

    if (ParentOp->Common.AmlOpcode == AML_PACKAGE_OP)
    {
        ParentOp = ParentOp->Common.Parent;
        if (!ParentOp)
        {
```

```
391             return (FALSE);
392         }

394         if (ParentOp->Common.AmlOpcode == AML_NAME_OP)
395         {
396             Node = ParentOp->Common.Node;

398             if (ACPI_COMPARE_NAME (Node->Name.Ascii, METHOD_NAME__PLD))
399             {
400                 return (TRUE);
401             }
402         }
403     }

405     return (FALSE);
406 }


409 /*******************************************************************************
410  *
411  * FUNCTION:    AcpiDmPldBuffer
412  *
413  * PARAMETERS:  Level               - Current source code indentation level
414  *              ByteData            - Pointer to the byte list
415  *              ByteCount           - Length of the byte list
416  *
417  * RETURN:      None
418  *
419  * DESCRIPTION: Dump and format the contents of a _PLD buffer object
420  *
421  ******************************************************************************/

423 #define ACPI_PLD_OUTPUT08     "%*.s/* %18s : %-6.2X */\n", ACPI_MUL_4 (Level), "
424 #define ACPI_PLD_OUTPUT16   "%*.s/* %18s : %-6.4X */\n", ACPI_MUL_4 (Level), " "
425 #define ACPI_PLD_OUTPUT24   "%*.s/* %18s : %-6.6X */\n", ACPI_MUL_4 (Level), " "

427 static void
428 AcpiDmPldBuffer (
429     UINT32                  Level,
430     UINT8                   *ByteData,
431     UINT32                  ByteCount)
432 {
433     ACPI_PLD_INFO           *PldInfo;
434     ACPI_STATUS             Status;


437     /* Check for valid byte count */

439     if (ByteCount < ACPI_PLD_REV1_BUFFER_SIZE)
440     {
441         return;
442     }

444     /* Convert _PLD buffer to local _PLD struct */

446     Status = AcpiDecodePldBuffer (ByteData, ByteCount, &PldInfo);
447     if (ACPI_FAILURE (Status))
448     {
449         return;
450     }

452     /* First 32-bit dword */

454     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Revision", PldInfo->Revision);
455     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "IgnoreColor", PldInfo->IgnoreColor);
456     AcpiOsPrintf (ACPI_PLD_OUTPUT24,"Color", PldInfo->Color);
```



```
458     /* Second 32-bit dword */

460     AcpiOsPrintf (ACPI_PLD_OUTPUT16,"Width", PldInfo->Width);
461     AcpiOsPrintf (ACPI_PLD_OUTPUT16,"Height", PldInfo->Height);

463     /* Third 32-bit dword */

465     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "UserVisible", PldInfo->UserVisible);
466     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Dock", PldInfo->Dock);
467     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Lid", PldInfo->Lid);
468     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Panel", PldInfo->Panel);
469     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "VerticalPosition", PldInfo->VerticalPosit
470     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "HorizontalPosition", PldInfo->HorizontalP
471     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Shape", PldInfo->Shape);
472     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "GroupOrientation", PldInfo->GroupOrientat
473     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "GroupToken", PldInfo->GroupToken);
474     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "GroupPosition", PldInfo->GroupPosition);
475     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Bay", PldInfo->Bay);

477     /* Fourth 32-bit dword */

479     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Ejectable", PldInfo->Ejectable);
480     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "OspmEjectRequired", PldInfo->OspmEjectReq
481     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "CabinetNumber", PldInfo->CabinetNumber);
482     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "CardCageNumber", PldInfo->CardCageNumber)
483     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Reference", PldInfo->Reference);
484     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Rotation", PldInfo->Rotation);
485     AcpiOsPrintf (ACPI_PLD_OUTPUT08,  "Order", PldInfo->Order);

487     /* Fifth 32-bit dword */

489     if (ByteCount >= ACPI_PLD_REV1_BUFFER_SIZE)
490     {
491         AcpiOsPrintf (ACPI_PLD_OUTPUT16,"VerticalOffset", PldInfo->VerticalOffse
492         AcpiOsPrintf (ACPI_PLD_OUTPUT16,"HorizontalOffset", PldInfo->HorizontalO
493     }

495     ACPI_FREE (PldInfo);
496 }


499 /*******************************************************************************
500  *
501  * FUNCTION:    AcpiDmUnicode
502  *
503  * PARAMETERS:  Op              - Byte List op containing Unicode string
504  *
505  * RETURN:      None
506  *
507  * DESCRIPTION: Dump Unicode string as a standard ASCII string. (Remove
508  *              the extra zero bytes).
509  *
510  ******************************************************************************/

512 static void
513 AcpiDmUnicode (
514     ACPI_PARSE_OBJECT       *Op)
515 {
516     UINT16                  *WordData;
517     UINT32                  WordCount;
518     UINT32                  i;


521     /* Extract the buffer info as a WORD buffer */
```

```
523     WordData = ACPI_CAST_PTR (UINT16, Op->Named.Data);
524     WordCount = ACPI_DIV_2 (((UINT32) Op->Common.Value.Integer));

526     /* Write every other byte as an ASCII character */

528     AcpiOsPrintf ("\"");
529     for (i = 0; i < (WordCount - 1); i++)
530     {
531         AcpiOsPrintf ("%c", (int) WordData[i]);
532     }

534     AcpiOsPrintf ("\")");
535 }


538 /*******************************************************************************
539  *
540  * FUNCTION:    AcpiDmIsEisaIdElement
541  *
542  * PARAMETERS:  Op              - Op to be examined
543  *
544  * RETURN:      None
545  *
546  * DESCRIPTION: Determine if an Op (argument to _HID or _CID) can be converted
547  *              to an EISA ID.
548  *
549  ******************************************************************************/

551 static void
552 AcpiDmIsEisaIdElement (
553     ACPI_PARSE_OBJECT       *Op)
554 {
555     UINT32                  BigEndianId;
556     UINT32                  Prefix[3];
557     UINT32                  i;


560     /* The parameter must be either a word or a dword */

562     if ((Op->Common.AmlOpcode != AML_DWORD_OP) &&
563         (Op->Common.AmlOpcode != AML_WORD_OP))
564     {
565         return;
566     }

568     /* Swap from little-endian to big-endian to simplify conversion */

570     BigEndianId = AcpiUtDwordByteSwap ((UINT32) Op->Common.Value.Integer);

572     /* Create the 3 leading ASCII letters */

574     Prefix[0] = ((BigEndianId >> 26) & 0x1F) + 0x40;
575     Prefix[1] = ((BigEndianId >> 21) & 0x1F) + 0x40;
576     Prefix[2] = ((BigEndianId >> 16) & 0x1F) + 0x40;

578     /* Verify that all 3 are ascii and alpha */

580     for (i = 0; i < 3; i++)
581     {
582         if (!ACPI_IS_ASCII (Prefix[i]) ||
583             !ACPI_IS_ALPHA (Prefix[i]))
584         {
585             return;
586         }
587     }
```



```
589     /* OK - mark this node as convertable to an EISA ID */

591     Op->Common.DisasmOpcode = ACPI_DASM_EISAID;
592 }


595 /*******************************************************************************
596  *
597  * FUNCTION:    AcpiDmIsEisaId
598  *
599  * PARAMETERS:  Op              - Op to be examined
600  *
601  * RETURN:      None
602  *
603  * DESCRIPTION: Determine if a Name() Op can be converted to an EisaId.
604  *
605  ******************************************************************************/

607 void
608 AcpiDmIsEisaId (
609     ACPI_PARSE_OBJECT       *Op)
610 {
611     UINT32                  Name;
612     ACPI_PARSE_OBJECT       *NextOp;


615     /* Get the NameSegment */

617     Name = AcpiPsGetName (Op);
618     if (!Name)
619     {
620         return;
621     }

623     NextOp = AcpiPsGetDepthNext (NULL, Op);
624     if (!NextOp)
625     {
626         return;
627     }

629     /* Check for _HID - has one argument */

631     if (ACPI_COMPARE_NAME (&Name, METHOD_NAME__HID))
632     {
633         AcpiDmIsEisaIdElement (NextOp);
634         return;
635     }

637     /* Exit if not _CID */

639     if (!ACPI_COMPARE_NAME (&Name, METHOD_NAME__CID))
640     {
641         return;
642     }

644     /* _CID can contain a single argument or a package */

646     if (NextOp->Common.AmlOpcode != AML_PACKAGE_OP)
647     {
648         AcpiDmIsEisaIdElement (NextOp);
649         return;
650     }

652     /* _CID with Package: get the package length */

654     NextOp = AcpiPsGetDepthNext (NULL, NextOp);
```

```
656     /* Don't need to use the length, just walk the peer list */

658     NextOp = NextOp->Common.Next;
659     while (NextOp)
660     {
661         AcpiDmIsEisaIdElement (NextOp);
662         NextOp = NextOp->Common.Next;
663     }
664 }


667 /*******************************************************************************
668  *
669  * FUNCTION:    AcpiDmEisaId
670  *
671  * PARAMETERS:  EncodedId       - Raw encoded EISA ID.
672  *
673  * RETURN:      None
674  *
675  * DESCRIPTION: Convert an encoded EISAID back to the original ASCII String.
676  *
677  ******************************************************************************/

679 void
680 AcpiDmEisaId (
681     UINT32                  EncodedId)
682 {
683     UINT32                  BigEndianId;


686     /* Swap from little-endian to big-endian to simplify conversion */

688     BigEndianId = AcpiUtDwordByteSwap (EncodedId);


691     /* Split to form "AAANNNN" string */

693     AcpiOsPrintf ("EisaId (\"%c%c%c%4.4X\")",

695         /* Three Alpha characters (AAA), 5 bits each */

697         (int) ((BigEndianId >> 26) & 0x1F) + 0x40,
698         (int) ((BigEndianId >> 21) & 0x1F) + 0x40,
699         (int) ((BigEndianId >> 16) & 0x1F) + 0x40,

701         /* Numeric part (NNNN) is simply the lower 16 bits */

703         (UINT32) (BigEndianId & 0xFFFF));
704 }

706 #endif
```

```
**********************************************************
    7921 Thu Dec 26 13:48:45 2013
new/usr/src/common/acpica/components/disassembler/dmdeferred.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmdeferred - Disassembly of deferred AML opcodes
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"
#include "amlcode.h"
#include "acdisasm.h"
#include "acparser.h"

#define _COMPONENT          ACPI_CA_DISASSEMBLER
        ACPI_MODULE_NAME    ("dmdeferred")


/* Local prototypes */

static ACPI_STATUS
AcpiDmDeferredParse (
    ACPI_PARSE_OBJECT       *Op,
    UINT8                   *Aml,
    UINT32                  AmlLength);


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmParseDeferredOps
 *
 * PARAMETERS:  Root                - Root of the parse tree
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Parse the deferred opcodes (Methods, regions, etc.)
 *
 ******************************************************************************/

ACPI_STATUS
AcpiDmParseDeferredOps (
    ACPI_PARSE_OBJECT       *Root)
{
    const ACPI_OPCODE_INFO  *OpInfo;
    ACPI_PARSE_OBJECT       *Op = Root;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_ENTRY ();


    /* Traverse the entire parse tree */

    while (Op)
    {
        OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
        if (!(OpInfo->Flags & AML_DEFER))
        {
            Op = AcpiPsGetDepthNext (Root, Op);
            continue;
        }

        /* Now we know we have a deferred opcode */

        switch (Op->Common.AmlOpcode)
        {
        case AML_METHOD_OP:
        case AML_BUFFER_OP:
        case AML_PACKAGE_OP:
        case AML_VAR_PACKAGE_OP:

            Status = AcpiDmDeferredParse (Op, Op->Named.Data, Op->Named.Length);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
            break;

        /* We don't need to do anything for these deferred opcodes */

        case AML_REGION_OP:
        case AML_DATA_REGION_OP:
        case AML_CREATE_QWORD_FIELD_OP:
        case AML_CREATE_DWORD_FIELD_OP:
        case AML_CREATE_WORD_FIELD_OP:
        case AML_CREATE_BYTE_FIELD_OP:
        case AML_CREATE_BIT_FIELD_OP:
        case AML_CREATE_FIELD_OP:
        case AML_BANK_FIELD_OP:
```

```
128             break;
130         default:
132             ACPI_ERROR ((AE_INFO, "Unhandled deferred AML opcode [0x%.4X]",
133                 Op->Common.AmlOpcode));
134             break;
135         }
137         Op = AcpiPsGetDepthNext (Root, Op);
138     }
140     return (AE_OK);
141 }


144 /*******************************************************************************
145  *
146  * FUNCTION:    AcpiDmDeferredParse
147  *
148  * PARAMETERS:  Op                  - Root Op of the deferred opcode
149  *              Aml                 - Pointer to the raw AML
150  *              AmlLength           - Length of the AML
151  *
152  * RETURN:      Status
153  *
154  * DESCRIPTION: Parse one deferred opcode
155  *              (Methods, operation regions, etc.)
156  *
157  ******************************************************************************/

159 static ACPI_STATUS
160 AcpiDmDeferredParse (
161     ACPI_PARSE_OBJECT       *Op,
162     UINT8                   *Aml,
163     UINT32                  AmlLength)
164 {
165     ACPI_WALK_STATE         *WalkState;
166     ACPI_STATUS             Status;
167     ACPI_PARSE_OBJECT       *SearchOp;
168     ACPI_PARSE_OBJECT       *StartOp;
169     UINT32                  BaseAmlOffset;
170     ACPI_PARSE_OBJECT       *NewRootOp;
171     ACPI_PARSE_OBJECT       *ExtraOp;


174     ACPI_FUNCTION_TRACE (DmDeferredParse);


177     if (!Aml || !AmlLength)
178     {
179         return_ACPI_STATUS (AE_OK);
180     }

182     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Parsing deferred opcode %s [%4.4s]\n",
183         Op->Common.AmlOpName, (char *) &Op->Named.Name));

185     /* Need a new walk state to parse the AML */

187     WalkState = AcpiDsCreateWalkState (0, Op, NULL, NULL);
188     if (!WalkState)
189     {
190         return_ACPI_STATUS (AE_NO_MEMORY);
191     }

193     Status = AcpiDsInitAmlWalk (WalkState, Op, NULL, Aml,
```



```
194         AmlLength, NULL, ACPI_IMODE_LOAD_PASS1);
195     if (ACPI_FAILURE (Status))
196     {
197         return_ACPI_STATUS (Status);
198     }

200     /* Parse the AML for this deferred opcode */

202     WalkState->ParseFlags &= ~ACPI_PARSE_DELETE_TREE;
203     WalkState->ParseFlags |= ACPI_PARSE_DISASSEMBLE;
204     Status = AcpiPsParseAml (WalkState);

206     /*
207      * We need to update all of the AML offsets, since the parser thought
208      * that the method began at offset zero. In reality, it began somewhere
209      * within the ACPI table, at the BaseAmlOffset. Walk the entire tree that
210      * was just created and update the AmlOffset in each Op.
211      */
212     BaseAmlOffset = (Op->Common.Value.Arg)->Common.AmlOffset + 1;
213     StartOp = (Op->Common.Value.Arg)->Common.Next;
214     SearchOp = StartOp;

216     while (SearchOp)
217     {
218         SearchOp->Common.AmlOffset += BaseAmlOffset;
219         SearchOp = AcpiPsGetDepthNext (StartOp, SearchOp);
220     }

222     /*
223      * For Buffer and Package opcodes, link the newly parsed subtree
224      * into the main parse tree
225      */
226     switch (Op->Common.AmlOpcode)
227     {
228     case AML_BUFFER_OP:
229     case AML_PACKAGE_OP:
230     case AML_VAR_PACKAGE_OP:

232         switch (Op->Common.AmlOpcode)
233         {
234         case AML_PACKAGE_OP:

236             ExtraOp = Op->Common.Value.Arg;
237             NewRootOp = ExtraOp->Common.Next;
238             ACPI_FREE (ExtraOp);
239             break;

241         case AML_VAR_PACKAGE_OP:
242         case AML_BUFFER_OP:
243         default:

245             NewRootOp = Op->Common.Value.Arg;
246             break;
247         }

249         Op->Common.Value.Arg = NewRootOp->Common.Value.Arg;

251         /* Must point all parents to the main tree */

253         StartOp = Op;
254         SearchOp = StartOp;
255         while (SearchOp)
256         {
257             if (SearchOp->Common.Parent == NewRootOp)
258             {
259                 SearchOp->Common.Parent = Op;
```

```
260             }

262             SearchOp = AcpiPsGetDepthNext (StartOp, SearchOp);
263         }

265         ACPI_FREE (NewRootOp);
266         break;

268     default:

270         break;
271     }

273     return_ACPI_STATUS (AE_OK);
274 }
```

****
11854 Thu Dec 26 13:48:45 2013
new/usr/src/common/acpica/components/disassembler/dmnames.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dmnames - AML disassembler, names, namestrings, pathnames
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acnamesp.h"
#include "acdisasm.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmnames")

/* Local prototypes */
```



```
#ifdef ACPI_OBSOLETE_FUNCTIONS
void
AcpiDmDisplayPath (
    ACPI_PARSE_OBJECT       *Op);
#endif


/******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpName
 *
 * PARAMETERS:  Name            - 4 character ACPI name
 *
 * RETURN:      Final length of name
 *
 * DESCRIPTION: Dump an ACPI name, minus any trailing underscores.
 *
 *****************************************************************************/

UINT32
AcpiDmDumpName (
    UINT32                  Name)
{
    UINT32                  i;
    UINT32                  Length;
    char                    NewName[4];


    /* Copy name locally in case the original name is not writeable */

    *ACPI_CAST_PTR (UINT32, &NewName[0]) = Name;

    /* Ensure that the name is printable, even if we have to fix it */

    AcpiUtRepairName (NewName);

    /* Remove all trailing underscores from the name */

    Length = ACPI_NAME_SIZE;
    for (i = (ACPI_NAME_SIZE - 1); i != 0; i--)
    {
        if (NewName[i] == '_')
        {
            Length--;
        }
        else
        {
            break;
        }
    }

    /* Dump the name, up to the start of the trailing underscores */

    for (i = 0; i < Length; i++)
    {
        AcpiOsPrintf ("%c", NewName[i]);
    }

    return (Length);
}
```

_____unchanged_portion_omitted_

```
/******************************************************************************
 *
 * FUNCTION:    AcpiDmNamestring
```

```
 205  *
 206  * PARAMETERS:  Name                - ACPI Name string to store
 207  *
 208  * RETURN:      None
 209  *
 210  * DESCRIPTION: Decode and dump an ACPI namestring. Handles prefix characters
 211  *
 212  *****************************************************************************/

 214 void
 215 AcpiDmNamestring (
 216     char                    *Name)
 217 {
 218     UINT32                  SegCount;


 221     if (!Name)
 222     {
 223         return;
 224     }

 226     /* Handle all Scope Prefix operators */

 228     while (ACPI_IS_ROOT_PREFIX (ACPI_GET8 (Name)) ||
 229             ACPI_IS_PARENT_PREFIX (ACPI_GET8 (Name)))
 229     while (AcpiPsIsPrefixChar (ACPI_GET8 (Name)))
 230     {
 231         /* Append prefix character */

 233         AcpiOsPrintf ("%1c", ACPI_GET8 (Name));
 234         Name++;
 235     }

 237     switch (ACPI_GET8 (Name))
 238     {
 239     case 0:


 241         SegCount = 0;
 242         break;

 244     case AML_DUAL_NAME_PREFIX:


 246         SegCount = 2;
 247         Name++;
 248         break;

 250     case AML_MULTI_NAME_PREFIX_OP:


 252         SegCount = (UINT32) ACPI_GET8 (Name + 1);
 253         Name += 2;
 254         break;

 256     default:


 258         SegCount = 1;
 259         break;
 260     }

 262     while (SegCount)
 263     {
 264         /* Append Name segment */

 266         AcpiDmDumpName (*ACPI_CAST_PTR (UINT32, Name));

 268         SegCount--;
 269         if (SegCount)
```



```
 270         {
 271             /* Not last name, append dot separator */

 273             AcpiOsPrintf (".");
 274         }
 275         Name += ACPI_NAME_SIZE;
 276     }
 277 }


 280 #ifdef ACPI_OBSOLETE_FUNCTIONS
 281 /*****************************************************************************
 282  *
 283  * FUNCTION:    AcpiDmDisplayPath
 284  *
 285  * PARAMETERS:  Op                  - Named Op whose path is to be constructed
 286  *
 287  * RETURN:      None
 288  *
 289  * DESCRIPTION: Walk backwards from current scope and display the name
 290  *              of each previous level of scope up to the root scope
 291  *              (like "pwd" does with file systems)
 292  *
 293  *****************************************************************************/

 295 void
 296 AcpiDmDisplayPath (
 297     ACPI_PARSE_OBJECT       *Op)
 298 {
 299     ACPI_PARSE_OBJECT       *Prev;
 300     ACPI_PARSE_OBJECT       *Search;
 301     UINT32                  Name;
 302     BOOLEAN                 DoDot = FALSE;
 303     ACPI_PARSE_OBJECT       *NamePath;
 304     const ACPI_OPCODE_INFO  *OpInfo;


 307     /* We are only interested in named objects */

 309     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
 310     if (!(OpInfo->Flags & AML_NSNODE))
 311     {
 312         return;
 313     }

 315     if (OpInfo->Flags & AML_CREATE)
 316     {
 317         /* Field creation - check for a fully qualified namepath */

 319         if (Op->Common.AmlOpcode == AML_CREATE_FIELD_OP)
 320         {
 321             NamePath = AcpiPsGetArg (Op, 3);
 322         }
 323         else
 324         {
 325             NamePath = AcpiPsGetArg (Op, 2);
 326         }

 328         if ((NamePath) &&
 329             (NamePath->Common.Value.String) &&
 330             (ACPI_IS_ROOT_PREFIX (NamePath->Common.Value.String[0])))
 326             (NamePath->Common.Value.String[0] == '\\'))
 331         {
 332             AcpiDmNamestring (NamePath->Common.Value.String);
 333             return;
 334         }
```

```
335     }
337     Prev = NULL;            /* Start with Root Node */
339     while (Prev != Op)
340     {
341         /* Search upwards in the tree to find scope with "prev" as its parent */
343         Search = Op;
344         for (; ;)
345         {
346             if (Search->Common.Parent == Prev)
347             {
348                 break;
349             }
351             /* Go up one level */
353             Search = Search->Common.Parent;
354         }
356         if (Prev)
357         {
358             OpInfo = AcpiPsGetOpcodeInfo (Search->Common.AmlOpcode);
359             if (!(OpInfo->Flags & AML_FIELD))
360             {
361                 /* Below root scope, append scope name */
363                 if (DoDot)
364                 {
365                     /* Append dot */
367                     AcpiOsPrintf (".");
368                 }
370                 if (OpInfo->Flags & AML_CREATE)
371                 {
372                     if (Op->Common.AmlOpcode == AML_CREATE_FIELD_OP)
373                     {
374                         NamePath = AcpiPsGetArg (Op, 3);
375                     }
376                     else
377                     {
378                         NamePath = AcpiPsGetArg (Op, 2);
379                     }
381                     if ((NamePath) &&
382                         (NamePath->Common.Value.String))
383                     {
384                         AcpiDmDumpName (NamePath->Common.Value.String);
385                     }
386                 }
387                 else
388                 {
389                     Name = AcpiPsGetName (Search);
390                     AcpiDmDumpName ((char *) &Name);
391                 }
393                 DoDot = TRUE;
394             }
395         }
396         Prev = Search;
397     }
398 }
```

*unchanged_portion_omitted_*

```
453 #endif
```



```
455 #endif
```

****
15696 Thu Dec 26 13:48:46 2013
new/usr/src/common/acpica/components/disassembler/dmobject.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dmobject - ACPI object decode and display
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acdisasm.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmnames")

/* Local prototypes */

static void
AcpiDmDecodeNode (
```



```
    ACPI_NAMESPACE_NODE     *Node);


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpMethodInfo
 *
 * PARAMETERS:  Status          - Method execution status
 *              WalkState       - Current state of the parse tree walk
 *              Op              - Executing parse op
 *
 * RETURN:      None
 *
 * DESCRIPTION: Called when a method has been aborted because of an error.
 *              Dumps the method execution stack, and the method locals/args,
 *              and disassembles the AML opcode that failed.
 *
 ******************************************************************************/

void
AcpiDmDumpMethodInfo (
    ACPI_STATUS             Status,
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Next;
    ACPI_THREAD_STATE       *Thread;
    ACPI_WALK_STATE         *NextWalkState;
    ACPI_NAMESPACE_NODE     *PreviousMethod = NULL;


    /* Ignore control codes, they are not errors */

    if ((Status & AE_CODE_MASK) == AE_CODE_CONTROL)
    {
        return;
    }

    /* We may be executing a deferred opcode */

    if (WalkState->DeferredNode)
    {
        AcpiOsPrintf ("Executing subtree for Buffer/Package/Region\n");
        return;
    }

    /*
     * If there is no Thread, we are not actually executing a method.
     * This can happen when the iASL compiler calls the interpreter
     * to perform constant folding.
     */
    Thread = WalkState->Thread;
    if (!Thread)
    {
        return;
    }

    /* Display exception and method name */

    AcpiOsPrintf ("\n**** Exception %s during execution of method ",
        AcpiFormatException (Status));
    AcpiNsPrintNodePathname (WalkState->MethodNode, NULL);

    /* Display stack of executing methods */

    AcpiOsPrintf ("\n\nMethod Execution Stack:\n");
```

```
126     NextWalkState = Thread->WalkStateList;

128     /* Walk list of linked walk states */

130     while (NextWalkState)
131     {
132         AcpiOsPrintf ("    Method [%4.4s] executing: ",
133                 AcpiUtGetNodeName (NextWalkState->MethodNode));

135         /* First method is the currently executing method */

137         if (NextWalkState == WalkState)
138         {
139             if (Op)
140             {
141                 /* Display currently executing ASL statement */

143                 Next = Op->Common.Next;
144                 Op->Common.Next = NULL;

146                 AcpiDmDisassemble (NextWalkState, Op, ACPI_UINT32_MAX);
147                 Op->Common.Next = Next;
148             }
149         }
150         else
151         {
152             /*
153              * This method has called another method
154              * NOTE: the method call parse subtree is already deleted at this
155              * point, so we cannot disassemble the method invocation.
156              */
157             AcpiOsPrintf ("Call to method ");
158             AcpiNsPrintNodePathname (PreviousMethod, NULL);
159         }

161         PreviousMethod = NextWalkState->MethodNode;
162         NextWalkState = NextWalkState->Next;
163         AcpiOsPrintf ("\n");
164     }

166     /* Display the method locals and arguments */

168     AcpiOsPrintf ("\n");
169     AcpiDmDisplayLocals (WalkState);
170     AcpiOsPrintf ("\n");
171     AcpiDmDisplayArguments (WalkState);
172     AcpiOsPrintf ("\n");
173 }


176 /*******************************************************************************
177  *
178  * FUNCTION:    AcpiDmDecodeInternalObject
179  *
180  * PARAMETERS:  ObjDesc         - Object to be displayed
181  *
182  * RETURN:      None
183  *
184  * DESCRIPTION: Short display of an internal object. Numbers/Strings/Buffers.
185  *
186  ******************************************************************************/

188 void
189 AcpiDmDecodeInternalObject (
190     ACPI_OPERAND_OBJECT     *ObjDesc)
191 {
```



```
192     UINT32                  i;


195     if (!ObjDesc)
196     {
197         AcpiOsPrintf (" Uninitialized");
198         return;
199     }

201     if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND)
202     {
203         AcpiOsPrintf (" %p [%s]", ObjDesc, AcpiUtGetDescriptorName (ObjDesc));
204         return;
205     }

207     AcpiOsPrintf (" %s", AcpiUtGetObjectTypeName (ObjDesc));

209     switch (ObjDesc->Common.Type)
210     {
211     case ACPI_TYPE_INTEGER:

213         AcpiOsPrintf (" %8.8X%8.8X",
214                 ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
215         break;


217     case ACPI_TYPE_STRING:

219         AcpiOsPrintf ("(%u) \"%.24s",
220                 ObjDesc->String.Length, ObjDesc->String.Pointer);

222         if (ObjDesc->String.Length > 24)
223         {
224             AcpiOsPrintf ("...");
225         }
226         else
227         {
228             AcpiOsPrintf ("\"");
229         }
230         break;


232     case ACPI_TYPE_BUFFER:

234         AcpiOsPrintf ("(%u)", ObjDesc->Buffer.Length);
235         for (i = 0; (i < 8) && (i < ObjDesc->Buffer.Length); i++)
236         {
237             AcpiOsPrintf (" %2.2X", ObjDesc->Buffer.Pointer[i]);
238         }
239         break;


241     default:

243         AcpiOsPrintf (" %p", ObjDesc);
244         break;
245     }
246 }


249 /*******************************************************************************
250  *
251  * FUNCTION:    AcpiDmDecodeNode
252  *
253  * PARAMETERS:  Node        - Object to be displayed
254  *
```

```
 * RETURN:      None
 *
 * DESCRIPTION: Short display of a namespace node
 *
 ******************************************************************************/

static void
AcpiDmDecodeNode (
    ACPI_NAMESPACE_NODE     *Node)
{

    AcpiOsPrintf ("<Node>            Name %4.4s",
            AcpiUtGetNodeName (Node));

    if (Node->Flags & ANOBJ_METHOD_ARG)
    {
        AcpiOsPrintf (" [Method Arg]");
    }
    if (Node->Flags & ANOBJ_METHOD_LOCAL)
    {
        AcpiOsPrintf (" [Method Local]");
    }

    switch (Node->Type)
    {
    /* These types have no attached object */

    case ACPI_TYPE_DEVICE:

        AcpiOsPrintf (" Device");
        break;

    case ACPI_TYPE_THERMAL:

        AcpiOsPrintf (" Thermal Zone");
        break;

    default:

        AcpiDmDecodeInternalObject (AcpiNsGetAttachedObject (Node));
        break;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDisplayInternalObject
 *
 * PARAMETERS:  ObjDesc         - Object to be displayed
 *              WalkState       - Current walk state
 *
 * RETURN:      None
 *
 * DESCRIPTION: Short display of an internal object
 *
 ******************************************************************************/

void
AcpiDmDisplayInternalObject (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_WALK_STATE         *WalkState)
{
    UINT8                   Type;
```



```
    AcpiOsPrintf ("%p ", ObjDesc);

    if (!ObjDesc)
    {
        AcpiOsPrintf ("<Null Object>\n");
        return;
    }

    /* Decode the object type */

    switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
    {
    case ACPI_DESC_TYPE_PARSER:

        AcpiOsPrintf ("<Parser>  ");
        break;


    case ACPI_DESC_TYPE_NAMED:

        AcpiDmDecodeNode ((ACPI_NAMESPACE_NODE *) ObjDesc);
        break;


    case ACPI_DESC_TYPE_OPERAND:

        Type = ObjDesc->Common.Type;
        if (Type > ACPI_TYPE_LOCAL_MAX)
        {
            AcpiOsPrintf (" Type %X [Invalid Type]", (UINT32) Type);
            return;
        }

        /* Decode the ACPI object type */

        switch (ObjDesc->Common.Type)
        {
        case ACPI_TYPE_LOCAL_REFERENCE:

            AcpiOsPrintf ("[%s] ", AcpiUtGetReferenceName (ObjDesc));

            /* Decode the refererence */

            switch (ObjDesc->Reference.Class)
            {
            case ACPI_REFCLASS_LOCAL:

                AcpiOsPrintf ("%X ", ObjDesc->Reference.Value);
                if (WalkState)
                {
                    ObjDesc = WalkState->LocalVariables
                                [ObjDesc->Reference.Value].Object;
                    AcpiOsPrintf ("%p", ObjDesc);
                    AcpiDmDecodeInternalObject (ObjDesc);
                }
                break;


            case ACPI_REFCLASS_ARG:

                AcpiOsPrintf ("%X ", ObjDesc->Reference.Value);
                if (WalkState)
                {
                    ObjDesc = WalkState->Arguments
                                [ObjDesc->Reference.Value].Object;
                    AcpiOsPrintf ("%p", ObjDesc);
```

```
384                     AcpiDmDecodeInternalObject (ObjDesc);
385                 }
386                 break;


388             case ACPI_REFCLASS_INDEX:

390                 switch (ObjDesc->Reference.TargetType)
391                 {
392                 case ACPI_TYPE_BUFFER_FIELD:

394                     AcpiOsPrintf ("%p", ObjDesc->Reference.Object);
395                     AcpiDmDecodeInternalObject (ObjDesc->Reference.Object);
396                     break;

398                 case ACPI_TYPE_PACKAGE:

400                     AcpiOsPrintf ("%p", ObjDesc->Reference.Where);
401                     if (!ObjDesc->Reference.Where)
402                     {
403                         AcpiOsPrintf (" Uninitialized WHERE pointer");
404                     }
405                     else
406                     {
407                         AcpiDmDecodeInternalObject (
408                             *(ObjDesc->Reference.Where));
409                     }
410                     break;

412                 default:

414                     AcpiOsPrintf ("Unknown index target type");
415                     break;
416                 }
417                 break;


419             case ACPI_REFCLASS_REFOF:

421                 if (!ObjDesc->Reference.Object)
422                 {
423                     AcpiOsPrintf ("Uninitialized reference subobject pointer");
424                     break;
425                 }

427                 /* Reference can be to a Node or an Operand object */

429                 switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc->Reference.Object))
430                 {
431                 case ACPI_DESC_TYPE_NAMED:
432                     AcpiDmDecodeNode (ObjDesc->Reference.Object);
433                     break;

435                 case ACPI_DESC_TYPE_OPERAND:
436                     AcpiDmDecodeInternalObject (ObjDesc->Reference.Object);
437                     break;

439                 default:
440                     break;
441                 }
442                 break;


444             case ACPI_REFCLASS_NAME:

446                 AcpiDmDecodeNode (ObjDesc->Reference.Node);
```



```
447                 break;


449             case ACPI_REFCLASS_DEBUG:
450             case ACPI_REFCLASS_TABLE:

452                 AcpiOsPrintf ("\n");
453                 break;


455             default:    /* Unknown reference class */

457                 AcpiOsPrintf ("%2.2X\n", ObjDesc->Reference.Class);
458                 break;
459             }
460             break;


462         default:

464             AcpiOsPrintf ("<Obj>            ");
465             AcpiDmDecodeInternalObject (ObjDesc);
466             break;
467         }
468         break;


470     default:

472         AcpiOsPrintf ("<Not a valid ACPI Object Descriptor> [%s]",
473             AcpiUtGetDescriptorName (ObjDesc));
474         break;
475     }

477     AcpiOsPrintf ("\n");
478 }
```

*unchanged_portion_omitted_*

```
577 #endif
```

```
**********************************************************
   20503 Thu Dec 26 13:48:46 2013
new/usr/src/common/acpica/components/disassembler/dmopcode.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmopcode - AML disassembler, specific AML opcodes
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdisasm.h"

#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmopcode")

/* Local prototypes */

static void
AcpiDmMatchKeyword (
    ACPI_PARSE_OBJECT       *Op);
```



```
/*******************************************************************************
 *
 * FUNCTION:    AcpiDmPredefinedDescription
 *
 * PARAMETERS:  Op              - Name() parse object
 *
 * RETURN:      None
 *
 * DESCRIPTION: Emit a description comment for a predefined ACPI name.
 *              Used for iASL compiler only.
 *
 ******************************************************************************/

void
AcpiDmPredefinedDescription (
    ACPI_PARSE_OBJECT       *Op)
{
#ifdef ACPI_ASL_COMPILER
    const AH_PREDEFINED_NAME    *Info;
    char                        *NameString;
    int                         LastCharIsDigit;
    int                         LastCharsAreHex;


    if (!Op)
    {
        return;
    }

    /* Ensure that the comment field is emitted only once */

    if (Op->Common.DisasmFlags & ACPI_PARSEOP_PREDEF_CHECKED)
    {
        return;
    }
    Op->Common.DisasmFlags |= ACPI_PARSEOP_PREDEF_CHECKED;

    /* Predefined name must start with an underscore */

    NameString = ACPI_CAST_PTR (char, &Op->Named.Name);
    if (NameString[0] != '_')
    {
        return;
    }

    /*
     * Check for the special ACPI names:
     * _ACd, _ALd, _EJd, _Exx, _Lxx, _Qxx, _Wxx, _T_a
     * (where d=decimal_digit, x=hex_digit, a=anything)
     *
     * Convert these to the generic name for table lookup.
     * Note: NameString is guaranteed to be upper case here.
     */
    LastCharIsDigit =
        (ACPI_IS_DIGIT (NameString[3]));    /* d */
    LastCharsAreHex =
        (ACPI_IS_XDIGIT (NameString[2]) &&  /* xx */
         ACPI_IS_XDIGIT (NameString[3]));

    switch (NameString[1])
    {
    case 'A':

        if ((NameString[2] == 'C') && (LastCharIsDigit))
        {
```

```
127             NameString = "_ACx";
128         }
129         else if ((NameString[2] == 'L') && (LastCharIsDigit))
130         {
131             NameString = "_ALx";
132         }
133         break;

135     case 'E':

137         if ((NameString[2] == 'J') && (LastCharIsDigit))
138         {
139             NameString = "_EJx";
140         }
141         else if (LastCharsAreHex)
142         {
143             NameString = "_Exx";
144         }
145         break;

147     case 'L':

149         if (LastCharsAreHex)
150         {
151             NameString = "_Lxx";
152         }
153         break;

155     case 'Q':

157         if (LastCharsAreHex)
158         {
159             NameString = "_Qxx";
160         }
161         break;

163     case 'T':

165         if (NameString[2] == '_')
166         {
167             NameString = "_T_x";
168         }
169         break;

171     case 'W':

173         if (LastCharsAreHex)
174         {
175             NameString = "_Wxx";
176         }
177         break;

179     default:

181         break;
182     }

184     /* Match the name in the info table */

186     for (Info = AslPredefinedInfo; Info->Name; Info++)
187     {
188         if (ACPI_COMPARE_NAME (NameString, Info->Name))
189         {
190             AcpiOsPrintf ("  // %4.4s: %s",
191                 NameString, ACPI_CAST_PTR (char, Info->Description));
192             return;
```



```
193         }
194     }

196 #endif
197     return;
198 }


201 /*******************************************************************************
202  *
203  * FUNCTION:    AcpiDmFieldPredefinedDescription
204  *
205  * PARAMETERS:  Op              - Parse object
206  *
207  * RETURN:      None
208  *
209  * DESCRIPTION: Emit a description comment for a resource descriptor tag
210  *              (which is a predefined ACPI name.) Used for iASL compiler only.
211  *
212  ******************************************************************************/

214 void
215 AcpiDmFieldPredefinedDescription (
216     ACPI_PARSE_OBJECT       *Op)
217 {
218 #ifdef ACPI_ASL_COMPILER
219     ACPI_PARSE_OBJECT       *IndexOp;
220     char                    *Tag;
221     const ACPI_OPCODE_INFO  *OpInfo;
222     const AH_PREDEFINED_NAME *Info;


225     if (!Op)
226     {
227         return;
228     }

230     /* Ensure that the comment field is emitted only once */

232     if (Op->Common.DisasmFlags & ACPI_PARSEOP_PREDEF_CHECKED)
233     {
234         return;
235     }
236     Op->Common.DisasmFlags |= ACPI_PARSEOP_PREDEF_CHECKED;

238     /*
239      * Op must be one of the Create* operators: CreateField, CreateBitField,
240      * CreateByteField, CreateWordField, CreateDwordField, CreateQwordField
241      */
242     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
243     if (!(OpInfo->Flags & AML_CREATE))
244     {
245         return;
246     }

248     /* Second argument is the Index argument */

250     IndexOp = Op->Common.Value.Arg;
251     IndexOp = IndexOp->Common.Next;

253     /* Index argument must be a namepath */

255     if (IndexOp->Common.AmlOpcode != AML_INT_NAMEPATH_OP)
256     {
257         return;
258     }
```

```
260     /* Major cheat: We previously put the Tag ptr in the Node field */

262     Tag = ACPI_CAST_PTR (char, IndexOp->Common.Node);
263     if (!Tag)
264     {
265         return;
266     }

268     /* Match the name in the info table */

270     for (Info = AslPredefinedInfo; Info->Name; Info++)
271     {
272         if (ACPI_COMPARE_NAME (Tag, Info->Name))
273         {
274             AcpiOsPrintf ("  // %4.4s: %s", Tag,
275                 ACPI_CAST_PTR (char, Info->Description));
276             return;
277         }
278     }

280 #endif
281     return;
282 }


285 /*******************************************************************************
286  *
287  * FUNCTION:    AcpiDmMethodFlags
288  *
289  * PARAMETERS:  Op              - Method Object to be examined
290  *
291  * RETURN:      None
292  *
293  * DESCRIPTION: Decode control method flags
294  *
295  ******************************************************************************/

297 void
298 AcpiDmMethodFlags (
299     ACPI_PARSE_OBJECT       *Op)
300 {
301     UINT32                  Flags;
302     UINT32                  Args;


305     /* The next Op contains the flags */

307     Op = AcpiPsGetDepthNext (NULL, Op);
308     Flags = (UINT8) Op->Common.Value.Integer;
309     Args = Flags & 0x07;

311     /* Mark the Op as completed */

313     Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;

315     /* 1) Method argument count */

317     AcpiOsPrintf (", %u, ", Args);

319     /* 2) Serialize rule */

321     if (!(Flags & 0x08))
322     {
323         AcpiOsPrintf ("Not");
324     }
```



```
326     AcpiOsPrintf ("Serialized");

328     /* 3) SyncLevel */

330     if (Flags & 0xF0)
331     {
332         AcpiOsPrintf (", %u", Flags >> 4);
333     }
334 }


337 /*******************************************************************************
338  *
339  * FUNCTION:    AcpiDmFieldFlags
340  *
341  * PARAMETERS:  Op              - Field Object to be examined
342  *
343  * RETURN:      None
344  *
345  * DESCRIPTION: Decode Field definition flags
346  *
347  ******************************************************************************/

349 void
350 AcpiDmFieldFlags (
351     ACPI_PARSE_OBJECT       *Op)
352 {
353     UINT32                  Flags;


356     Op = Op->Common.Next;
357     Flags = (UINT8) Op->Common.Value.Integer;

359     /* Mark the Op as completed */

361     Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;

363     AcpiOsPrintf ("%s, ", AcpiGbl_AccessTypes [Flags & 0x07]);
364     AcpiOsPrintf ("%s, ", AcpiGbl_LockRule [(Flags & 0x10) >> 4]);
365     AcpiOsPrintf ("%s)",  AcpiGbl_UpdateRules [(Flags & 0x60) >> 5]);
366 }


369 /*******************************************************************************
370  *
371  * FUNCTION:    AcpiDmAddressSpace
372  *
373  * PARAMETERS:  SpaceId         - ID to be translated
374  *
375  * RETURN:      None
376  *
377  * DESCRIPTION: Decode a SpaceId to an AddressSpaceKeyword
378  *
379  ******************************************************************************/

381 void
382 AcpiDmAddressSpace (
383     UINT8                   SpaceId)
384 {

386     if (SpaceId >= ACPI_NUM_PREDEFINED_REGIONS)
387     {
388         if (SpaceId == 0x7F)
389         {
390             AcpiOsPrintf ("FFixedHW, ");
```

```
391         }
392         else
393         {
394             AcpiOsPrintf ("0x%.2X, ", SpaceId);
395         }
396     }
397     else
398     {
399         AcpiOsPrintf ("%s, ", AcpiGbl_RegionTypes [SpaceId]);
400     }
401 }


404 /*******************************************************************************
405  *
406  * FUNCTION:    AcpiDmRegionFlags
407  *
408  * PARAMETERS:  Op              - Object to be examined
409  *
410  * RETURN:      None
411  *
412  * DESCRIPTION: Decode OperationRegion flags
413  *
414  ******************************************************************************/

416 void
417 AcpiDmRegionFlags (
418     ACPI_PARSE_OBJECT       *Op)
419 {


422     /* The next Op contains the SpaceId */

424     Op = AcpiPsGetDepthNext (NULL, Op);

426     /* Mark the Op as completed */

428     Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;

430     AcpiOsPrintf (", ");
431     AcpiDmAddressSpace ((UINT8) Op->Common.Value.Integer);
432 }


435 /*******************************************************************************
436  *
437  * FUNCTION:    AcpiDmMatchOp
438  *
439  * PARAMETERS:  Op              - Match Object to be examined
440  *
441  * RETURN:      None
442  *
443  * DESCRIPTION: Decode Match opcode operands
444  *
445  ******************************************************************************/

447 void
448 AcpiDmMatchOp (
449     ACPI_PARSE_OBJECT       *Op)
450 {
451     ACPI_PARSE_OBJECT       *NextOp;


454     NextOp = AcpiPsGetDepthNext (NULL, Op);
455     NextOp = NextOp->Common.Next;
```



```
457     if (!NextOp)
458     {
459         /* Handle partial tree during single-step */

461         return;
462     }

464     /* Mark the two nodes that contain the encoding for the match keywords */

466     NextOp->Common.DisasmOpcode = ACPI_DASM_MATCHOP;

468     NextOp = NextOp->Common.Next;
469     NextOp = NextOp->Common.Next;
470     NextOp->Common.DisasmOpcode = ACPI_DASM_MATCHOP;
471 }


474 /*******************************************************************************
475  *
476  * FUNCTION:    AcpiDmMatchKeyword
477  *
478  * PARAMETERS:  Op              - Match Object to be examined
479  *
480  * RETURN:      None
481  *
482  * DESCRIPTION: Decode Match opcode operands
483  *
484  ******************************************************************************/

486 static void
487 AcpiDmMatchKeyword (
488     ACPI_PARSE_OBJECT       *Op)
489 {


492     if (((UINT32) Op->Common.Value.Integer) > ACPI_MAX_MATCH_OPCODE)
493     {
494         AcpiOsPrintf ("/* Unknown Match Keyword encoding */");
495     }
496     else
497     {
498         AcpiOsPrintf ("%s", ACPI_CAST_PTR (char,
499             AcpiGbl_MatchOps[(ACPI_SIZE) Op->Common.Value.Integer]));
500     }
501 }


504 /*******************************************************************************
505  *
506  * FUNCTION:    AcpiDmDisassembleOneOp
507  *
508  * PARAMETERS:  WalkState           - Current walk info
509  *              Info                - Parse tree walk info
510  *              Op                  - Op that is to be printed
511  *
512  * RETURN:      None
513  *
514  * DESCRIPTION: Disassemble a single AML opcode
515  *
516  ******************************************************************************/

518 void
519 AcpiDmDisassembleOneOp (
520     ACPI_WALK_STATE         *WalkState,
521     ACPI_OP_WALK_INFO       *Info,
522     ACPI_PARSE_OBJECT       *Op)
```

```
{
    const ACPI_OPCODE_INFO  *OpInfo = NULL;
    UINT32                  Offset;
    UINT32                  Length;
    ACPI_PARSE_OBJECT       *Child;
    ACPI_STATUS             Status;
    UINT8                   *Aml;


    if (!Op)
    {
        AcpiOsPrintf ("<NULL OP PTR>");
        return;
    }

    switch (Op->Common.DisasmOpcode)
    {
    case ACPI_DASM_MATCHOP:

        AcpiDmMatchKeyword (Op);
        return;

    case ACPI_DASM_LNOT_SUFFIX:

        switch (Op->Common.AmlOpcode)
        {
        case AML_LEQUAL_OP:

            AcpiOsPrintf ("LNotEqual");
            break;

        case AML_LGREATER_OP:

            AcpiOsPrintf ("LLessEqual");
            break;

        case AML_LLESS_OP:

            AcpiOsPrintf ("LGreaterEqual");
            break;

        default:

            break;
        }
        Op->Common.DisasmOpcode = 0;
        Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
        return;

    default:
        break;
    }


    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    /* The op and arguments */

    switch (Op->Common.AmlOpcode)
    {
    case AML_LNOT_OP:

        Child = Op->Common.Value.Arg;
        if ((Child->Common.AmlOpcode == AML_LEQUAL_OP) ||
            (Child->Common.AmlOpcode == AML_LGREATER_OP) ||
            (Child->Common.AmlOpcode == AML_LLESS_OP))
```



```
        {
            Child->Common.DisasmOpcode = ACPI_DASM_LNOT_SUFFIX;
            Op->Common.DisasmOpcode = ACPI_DASM_LNOT_PREFIX;
        }
        else
        {
            AcpiOsPrintf ("%s", OpInfo->Name);
        }
        break;

    case AML_BYTE_OP:

        AcpiOsPrintf ("0x%2.2X", (UINT32) Op->Common.Value.Integer);
        break;

    case AML_WORD_OP:

        if (Op->Common.DisasmOpcode == ACPI_DASM_EISAID)
        {
            AcpiDmEisaId ((UINT32) Op->Common.Value.Integer);
        }
        else
        {
            AcpiOsPrintf ("0x%4.4X", (UINT32) Op->Common.Value.Integer);
        }
        break;

    case AML_DWORD_OP:

        if (Op->Common.DisasmOpcode == ACPI_DASM_EISAID)
        {
            AcpiDmEisaId ((UINT32) Op->Common.Value.Integer);
        }
        else
        {
            AcpiOsPrintf ("0x%8.8X", (UINT32) Op->Common.Value.Integer);
        }
        break;

    case AML_QWORD_OP:

        AcpiOsPrintf ("0x%8.8X%8.8X",
            ACPI_FORMAT_UINT64 (Op->Common.Value.Integer));
        break;

    case AML_STRING_OP:

        AcpiUtPrintString (Op->Common.Value.String, ACPI_UINT16_MAX);
        break;

    case AML_BUFFER_OP:
        /*
         * Determine the type of buffer. We can have one of the following:
         *
         * 1) ResourceTemplate containing Resource Descriptors.
         * 2) Unicode String buffer
         * 3) ASCII String buffer
         * 4) Raw data buffer (if none of the above)
         *
         * Since there are no special AML opcodes to differentiate these
         * types of buffers, we have to closely look at the data in the
         * buffer to determine the type.
         */
        if (!AcpiGbl_NoResourceDisassembly)
        {
            Status = AcpiDmIsResourceTemplate (WalkState, Op);
```

```
655             if (ACPI_SUCCESS (Status))
656             {
657                 Op->Common.DisasmOpcode = ACPI_DASM_RESOURCE;
658                 AcpiOsPrintf ("ResourceTemplate");
659                 break;
660             }
661             else if (Status == AE_AML_NO_RESOURCE_END_TAG)
662             {
663                 AcpiOsPrintf ("/**** Is ResourceTemplate, but EndTag not at buff
664             }
665         }

667         if (AcpiDmIsUnicodeBuffer (Op))
668         {
669             Op->Common.DisasmOpcode = ACPI_DASM_UNICODE;
670             AcpiOsPrintf ("Unicode (");
671         }
672         else if (AcpiDmIsStringBuffer (Op))
673         {
674             Op->Common.DisasmOpcode = ACPI_DASM_STRING;
675             AcpiOsPrintf ("Buffer");
676         }
677         else if (AcpiDmIsPldBuffer (Op))
678         {
679             Op->Common.DisasmOpcode = ACPI_DASM_PLD_METHOD;
680             AcpiOsPrintf ("Buffer");
681         }
682         else
683         {
684             Op->Common.DisasmOpcode = ACPI_DASM_BUFFER;
685             AcpiOsPrintf ("Buffer");
686         }
687         break;

689     case AML_INT_STATICSTRING_OP:

691         if (Op->Common.Value.String)
692         {
693             AcpiOsPrintf ("%s", Op->Common.Value.String);
694         }
695         else
696         {
697             AcpiOsPrintf ("\"<NULL STATIC STRING PTR>\"");
698         }
699         break;

701     case AML_INT_NAMEPATH_OP:

703         AcpiDmNamestring (Op->Common.Value.Name);
704         break;

706     case AML_INT_NAMEDFIELD_OP:

708         Length = AcpiDmDumpName (Op->Named.Name);
709         AcpiOsPrintf (",%*.s  %u", (unsigned) (5 - Length), " ",
710             (UINT32) Op->Common.Value.Integer);
711         AcpiDmCommaIfFieldMember (Op);

713         Info->BitOffset += (UINT32) Op->Common.Value.Integer;
714         break;

716     case AML_INT_RESERVEDFIELD_OP:

718         /* Offset() -- Must account for previous offsets */

720         Offset = (UINT32) Op->Common.Value.Integer;
```



```
721         Info->BitOffset += Offset;

723         if (Info->BitOffset % 8 == 0)
724         {
725             AcpiOsPrintf ("Offset (0x%.2X)", ACPI_DIV_8 (Info->BitOffset));
726         }
727         else
728         {
729             AcpiOsPrintf ("    ,   %u", Offset);
730         }

732         AcpiDmCommaIfFieldMember (Op);
733         break;

735     case AML_INT_ACCESSFIELD_OP:
736     case AML_INT_EXTACCESSFIELD_OP:

738         AcpiOsPrintf ("AccessAs (%s, ",
739             AcpiGbl_AccessTypes [(UINT32) (Op->Common.Value.Integer & 0x7)]);

741         AcpiDmDecodeAttribute ((UINT8) (Op->Common.Value.Integer >> 8));

743         if (Op->Common.AmlOpcode == AML_INT_EXTACCESSFIELD_OP)
744         {
745             AcpiOsPrintf (" (0x%2.2X)", (unsigned) ((Op->Common.Value.Integer >>
746         }

748         AcpiOsPrintf (")");
749         AcpiDmCommaIfFieldMember (Op);
750         break;

752     case AML_INT_CONNECTION_OP:
753         /*
754          * Two types of Connection() - one with a buffer object, the
755          * other with a namestring that points to a buffer object.
756          */
757         AcpiOsPrintf ("Connection (");
758         Child = Op->Common.Value.Arg;

760         if (Child->Common.AmlOpcode == AML_INT_BYTELIST_OP)
761         {
762             AcpiOsPrintf ("\n");

764             Aml = Child->Named.Data;
765             Length = (UINT32) Child->Common.Value.Integer;

767             Info->Level += 1;
768             Op->Common.DisasmOpcode = ACPI_DASM_RESOURCE;
769             AcpiDmResourceTemplate (Info, Op->Common.Parent, Aml, Length);

771             Info->Level -= 1;
772             AcpiDmIndent (Info->Level);
773         }
774         else
775         {
776             AcpiDmNamestring (Child->Common.Value.Name);
777         }

779         AcpiOsPrintf (")");
780         AcpiDmCommaIfFieldMember (Op);
781         AcpiOsPrintf ("\n");

783         Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE; /* for now, ignore in Acp
784         Child->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
785         break;
```

```
787     case AML_INT_BYTELIST_OP:

789         AcpiDmByteList (Info, Op);
790         break;

792     case AML_INT_METHODCALL_OP:

794         Op = AcpiPsGetDepthNext (NULL, Op);
795         Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;

797         AcpiDmNamestring (Op->Common.Value.Name);
798         break;

800     default:

802         /* Just get the opcode name and print it */

804         AcpiOsPrintf ("%s", OpInfo->Name);


807 #ifdef ACPI_DEBUGGER

809         if ((Op->Common.AmlOpcode == AML_INT_RETURN_VALUE_OP) &&
810             (WalkState) &&
811             (WalkState->Results) &&
812             (WalkState->ResultCount))
813         {
814             AcpiDmDecodeInternalObject (
815                 WalkState->Results->Results.ObjDesc [
816                     (WalkState->ResultCount - 1) %
817                         ACPI_RESULTS_FRAME_OBJ_NUM]);
818         }
819 #endif

821         break;
822     }
823 }

825 #endif  /* ACPI_DISASSEMBLER */
```

****
13510 Thu Dec 26 13:48:46 2013
new/usr/src/common/acpica/components/disassembler/dmresrc.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dmresrc.c - Resource Descriptor disassembly
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acdisasm.h"

#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbresrc")


/* Dispatch tables for Resource disassembly functions */

typedef
void (*ACPI_RESOURCE_HANDLER) (
```



```
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level);

static ACPI_RESOURCE_HANDLER    AcpiGbl_DmResourceDispatch [] =
{
    /* Small descriptors */

    NULL,                           /* 0x00, Reserved */
    NULL,                           /* 0x01, Reserved */
    NULL,                           /* 0x02, Reserved */
    NULL,                           /* 0x03, Reserved */
    AcpiDmIrqDescriptor,            /* 0x04, ACPI_RESOURCE_NAME_IRQ_FORMAT */
    AcpiDmDmaDescriptor,            /* 0x05, ACPI_RESOURCE_NAME_DMA_FORMAT */
    AcpiDmStartDependentDescriptor, /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT
    AcpiDmEndDependentDescriptor,   /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
    AcpiDmIoDescriptor,             /* 0x08, ACPI_RESOURCE_NAME_IO_PORT */
    AcpiDmFixedIoDescriptor,        /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO_PORT */
    AcpiDmFixedDmaDescriptor,       /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
    NULL,                           /* 0x0A, Reserved */
    NULL,                           /* 0x0B, Reserved */
    NULL,                           /* 0x0C, Reserved */
    NULL,                           /* 0x0D, Reserved */
    AcpiDmVendorSmallDescriptor,    /* 0x0E, ACPI_RESOURCE_NAME_SMALL_VENDOR */
    NULL,                           /* 0x0F, ACPI_RESOURCE_NAME_END_TAG (not use

    /* Large descriptors */

    NULL,                           /* 0x00, Reserved */
    AcpiDmMemory24Descriptor,       /* 0x01, ACPI_RESOURCE_NAME_MEMORY_24 */
    AcpiDmGenericRegisterDescriptor,/* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER
    NULL,                           /* 0x03, Reserved */
    AcpiDmVendorLargeDescriptor,    /* 0x04, ACPI_RESOURCE_NAME_LARGE_VENDOR */
    AcpiDmMemory32Descriptor,       /* 0x05, ACPI_RESOURCE_NAME_MEMORY_32 */
    AcpiDmFixedMemory32Descriptor,  /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY_32
    AcpiDmDwordDescriptor,          /* 0x07, ACPI_RESOURCE_NAME_DWORD_ADDRESS_SP
    AcpiDmWordDescriptor,           /* 0x08, ACPI_RESOURCE_NAME_WORD_ADDRESS_SPA
    AcpiDmInterruptDescriptor,      /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_XRUPT *
    AcpiDmQwordDescriptor,          /* 0x0A, ACPI_RESOURCE_NAME_QWORD_ADDRESS_SP
    AcpiDmExtendedDescriptor,       /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
    AcpiDmGpioDescriptor,           /* 0x0C, ACPI_RESOURCE_NAME_GPIO */
    NULL,                           /* 0x0D, Reserved */
    AcpiDmSerialBusDescriptor       /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS */
    AcpiDmExtendedDescriptor        /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
};
```

*unchanged_portion_omitted_*

```
/******************************************************************************
 *
 * FUNCTION:    AcpiDmResourceTemplate
 *
 * PARAMETERS:  Info            - Curent parse tree walk info
 *              ByteData        - Pointer to the byte list data
 *              ByteCount       - Length of the byte list
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump the contents of a Resource Template containing a set of
 *              Resource Descriptors.
 *
 *****************************************************************************/

void
AcpiDmResourceTemplate (
    ACPI_OP_WALK_INFO       *Info,
```

```
248     ACPI_PARSE_OBJECT       *Op,
249     UINT8                   *ByteData,
250     UINT32                  ByteCount)
251 {
252     ACPI_STATUS             Status;
253     UINT32                  CurrentByteOffset;
254     UINT8                   ResourceType;
255     UINT32                  ResourceLength;
256     void                    *Aml;
257     UINT32                  Level;
258     BOOLEAN                 DependentFns = FALSE;
259     UINT8                   ResourceIndex;
260     ACPI_NAMESPACE_NODE     *Node;


263     Level = Info->Level;
264     ResourceName = ACPI_DEFAULT_RESNAME;
265     Node = Op->Common.Node;
266     if (Node)
267     {
268         Node = Node->Child;
269     }

271     for (CurrentByteOffset = 0; CurrentByteOffset < ByteCount;)
272     {
273         Aml = &ByteData[CurrentByteOffset];

275         /* Get the descriptor type and length */

277         ResourceType = AcpiUtGetResourceType (Aml);
278         ResourceLength = AcpiUtGetResourceLength (Aml);

280         /* Validate the Resource Type and Resource Length */

282         Status = AcpiUtValidateResource (NULL, Aml, &ResourceIndex);
285         Status = AcpiUtValidateResource (Aml, &ResourceIndex);
283         if (ACPI_FAILURE (Status))
284         {
285             AcpiOsPrintf ("/*** Could not validate Resource, type (%X) %s***/\n"
286                 ResourceType, AcpiFormatException (Status));
287             return;
288         }

290         /* Point to next descriptor */

292         CurrentByteOffset += AcpiUtGetDescriptorLength (Aml);

294         /* Descriptor pre-processing */

296         switch (ResourceType)
297         {
298         case ACPI_RESOURCE_NAME_START_DEPENDENT:

300             /* Finish a previous StartDependentFns */

302             if (DependentFns)
303             {
304                 Level--;
305                 AcpiDmIndent (Level);
306                 AcpiOsPrintf ("}\n");
307             }
308             break;

310         case ACPI_RESOURCE_NAME_END_DEPENDENT:

312             Level--;
```



```
313             DependentFns = FALSE;
314             break;

316         case ACPI_RESOURCE_NAME_END_TAG:

318             /* Normal exit, the resource list is finished */

320             if (DependentFns)
321             {
322                 /*
323                  * Close an open StartDependentDescriptor. This indicates a
324                  * missing EndDependentDescriptor.
325                  */
326                 Level--;
327                 DependentFns = FALSE;

329                 /* Go ahead and insert EndDependentFn() */

331                 AcpiDmEndDependentDescriptor (Aml, ResourceLength, Level);

333                 AcpiDmIndent (Level);
334                 AcpiOsPrintf (
335                     "/*** Disassembler: inserted missing EndDependentFn () ***/\
336             }
337             return;

339         default:

341             break;
342         }

344         /* Disassemble the resource structure */

346         if (Node)
347         {
348             ResourceName = Node->Name.Integer;
349             Node = Node->Peer;
350         }

352         AcpiGbl_DmResourceDispatch [ResourceIndex] (
353             Aml, ResourceLength, Level);

355         /* Descriptor post-processing */

357         if (ResourceType == ACPI_RESOURCE_NAME_START_DEPENDENT)
358         {
359             DependentFns = TRUE;
360             Level++;
361         }
362     }
363 }


366 /*******************************************************************************
367  *
368  * FUNCTION:    AcpiDmIsResourceTemplate
369  *
370  * PARAMETERS:  WalkState           - Current walk info
371  *              Op                  - Buffer Op to be examined
372  * PARAMETERS:  Op          - Buffer Op to be examined
372  *
373  * RETURN:      Status. AE_OK if valid template
374  *
375  * DESCRIPTION: Walk a byte list to determine if it consists of a valid set
376  *              of resource descriptors. Nothing is output.
377  *
```

```
 378  *****************************************************************************/

 380 ACPI_STATUS
 381 AcpiDmIsResourceTemplate (
 382     ACPI_WALK_STATE         *WalkState,
 383     ACPI_PARSE_OBJECT       *Op)
 384 {
 385     ACPI_STATUS             Status;
 386     ACPI_PARSE_OBJECT       *NextOp;
 387     UINT8                   *Aml;
 388     UINT8                   *EndAml;
 389     ACPI_SIZE               Length;


 392     /* This op must be a buffer */

 394     if (Op->Common.AmlOpcode != AML_BUFFER_OP)
 395     {
 396         return (AE_TYPE);
 397     }

 399     /* Get the ByteData list and length */

 401     NextOp = Op->Common.Value.Arg;
 402     if (!NextOp)
 403     {
 404         AcpiOsPrintf ("NULL byte list in buffer\n");
 405         return (AE_TYPE);
 406     }

 408     NextOp = NextOp->Common.Next;
 409     if (!NextOp)
 410     {
 411         return (AE_TYPE);
 412     }

 414     Aml = NextOp->Named.Data;
 415     Length = (ACPI_SIZE) NextOp->Common.Value.Integer;

 417     /* Walk the byte list, abort on any invalid descriptor type or length */

 419     Status = AcpiUtWalkAmlResources (WalkState, Aml, Length,
 420         NULL, ACPI_CAST_INDIRECT_PTR (void, &EndAml));
 413     Status = AcpiUtWalkAmlResources (Aml, Length, NULL, &EndAml);
 421     if (ACPI_FAILURE (Status))
 422     {
 423         return (AE_TYPE);
 424     }

 426     /*
 427      * For the resource template to be valid, one EndTag must appear
 428      * at the very end of the ByteList, not before. (For proper disassembly
 429      * of a ResourceTemplate, the buffer must not have any extra data after
 430      * the EndTag.)
 431      */
 432     if ((Aml + Length - sizeof (AML_RESOURCE_END_TAG)) != EndAml)
 433     {
 434         return (AE_AML_NO_RESOURCE_END_TAG);
 435     }

 437     /*
 438      * All resource descriptors are valid, therefore this list appears
 439      * to be a valid resource template
 440      */
 441     return (AE_OK);
 442 }
```

*unchanged_portion_omitted_*

****
29253 Thu Dec 26 13:48:47 2013
new/usr/src/common/acpica/components/disassembler/dmresrcl.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dmresrcl.c - "Large" Resource Descriptor disassembly
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbresrcl")


/* Common names for address and memory descriptors */

static char                 *AcpiDmAddressNames[] =
{
```



```
    "Granularity",
    "Range Minimum",
    "Range Maximum",
    "Translation Offset",
    "Length"
};
```

*_____unchanged_portion_omitted_*

```
/* Local prototypes */

static void
AcpiDmSpaceFlags (
        UINT8               Flags);

static void
AcpiDmIoFlags (
        UINT8               Flags);

static void
AcpiDmIoFlags2 (
        UINT8               SpecificFlags);

static void
AcpiDmMemoryFlags (
    UINT8                   Flags,
    UINT8                   SpecificFlags);

static void
AcpiDmMemoryFlags2 (
    UINT8                   SpecificFlags);

static void
AcpiDmResourceSource (
    AML_RESOURCE            *Resource,
    ACPI_SIZE               MinimumLength,
    UINT32                  Length);

static void
AcpiDmAddressFields (
    void                    *Source,
    UINT8                   Type,
    UINT32                  Level);

static void
AcpiDmAddressPrefix (
    UINT8                   Type);

static void
AcpiDmAddressCommon (
    AML_RESOURCE            *Resource,
    UINT8                   Type,
    UINT32                  Level);

static void
AcpiDmAddressFlags (
    AML_RESOURCE            *Resource);


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmMemoryFields
 *
 * PARAMETERS:  Source              - Pointer to the contiguous data fields
 *              Type                - 16 or 32 (bit)
 *              Level               - Current source code indentation level
```

```
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode fields common to Memory24 and Memory32 descriptors
 *
 ******************************************************************************/

static void
AcpiDmMemoryFields (
    void                    *Source,
    UINT8                   Type,
    UINT32                  Level)
{
    UINT32                  i;


    for (i = 0; i < 4; i++)
    {
        AcpiDmIndent (Level + 1);

        switch (Type)
        {
        case 16:

            AcpiDmDumpInteger16 (ACPI_CAST_PTR (UINT16, Source)[i],
                AcpiDmMemoryNames[i]);
            break;

        case 32:

            AcpiDmDumpInteger32 (ACPI_CAST_PTR (UINT32, Source)[i],
                AcpiDmMemoryNames[i]);
            break;

        default:

            return;
        }
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmAddressFields
 *
 * PARAMETERS:  Source              - Pointer to the contiguous data fields
 *              Type                - 16, 32, or 64 (bit)
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode fields common to address descriptors
 *
 ******************************************************************************/

static void
AcpiDmAddressFields (
    void                    *Source,
    UINT8                   Type,
    UINT32                  Level)
{
    UINT32                  i;


    AcpiOsPrintf ("\n");
```



```
    for (i = 0; i < 5; i++)
    {
        AcpiDmIndent (Level + 1);

        switch (Type)
        {
        case 16:

            AcpiDmDumpInteger16 (ACPI_CAST_PTR (UINT16, Source)[i],
                AcpiDmAddressNames[i]);
            break;

        case 32:

            AcpiDmDumpInteger32 (ACPI_CAST_PTR (UINT32, Source)[i],
                AcpiDmAddressNames[i]);
            break;

        case 64:

            AcpiDmDumpInteger64 (ACPI_CAST_PTR (UINT64, Source)[i],
                AcpiDmAddressNames[i]);
            break;

        default:

            return;
        }
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmAddressPrefix
 *
 * PARAMETERS:  Type                - Descriptor type
 *
 * RETURN:      None
 *
 * DESCRIPTION: Emit name prefix representing the address descriptor type
 *
 ******************************************************************************/

static void
AcpiDmAddressPrefix (
    UINT8                   Type)
{

    switch (Type)
    {
    case ACPI_RESOURCE_TYPE_ADDRESS16:

        AcpiOsPrintf ("Word");
        break;

    case ACPI_RESOURCE_TYPE_ADDRESS32:

        AcpiOsPrintf ("DWord");
        break;

    case ACPI_RESOURCE_TYPE_ADDRESS64:

        AcpiOsPrintf ("QWord");
        break;
```

```
266     case ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64:

268         AcpiOsPrintf ("Extended");
269         break;

271     default:

273         return;
274     }
275 }


278 /*******************************************************************************
279  *
280  * FUNCTION:    AcpiDmAddressCommon
281  *
282  * PARAMETERS:  Resource            - Raw AML descriptor
283  *              Type                - Descriptor type
284  *              Level               - Current source code indentation level
285  *
286  * RETURN:      None
287  *
288  * DESCRIPTION: Emit common name and flag fields common to address descriptors
289  *
290  ******************************************************************************/

292 static void
293 AcpiDmAddressCommon (
294     AML_RESOURCE            *Resource,
295     UINT8                   Type,
296     UINT32                  Level)
297 {
298     UINT8                   ResourceType;
299     UINT8                   SpecificFlags;
300     UINT8                   Flags;


303     ResourceType = Resource->Address.ResourceType;
304     SpecificFlags = Resource->Address.SpecificFlags;
305     Flags = Resource->Address.Flags;

307     AcpiDmIndent (Level);

309     /* Validate ResourceType */

311     if ((ResourceType > 2) && (ResourceType < 0xC0))
312     {
313         AcpiOsPrintf ("/**** Invalid Resource Type: 0x%X ****/", ResourceType);
314         return;
315     }

317     /* Prefix is either Word, DWord, QWord, or Extended */

319     AcpiDmAddressPrefix (Type);

321     /* Resource Types above 0xC0 are vendor-defined */

323     if (ResourceType > 2)
324     {
325         AcpiOsPrintf ("Space (0x%2.2X, ", ResourceType);
326         AcpiDmSpaceFlags (Flags);
327         AcpiOsPrintf (" 0x%2.2X,", SpecificFlags);
328         return;
329     }
```



```
331     /* This is either a Memory, IO, or BusNumber descriptor (0,1,2) */

333     AcpiOsPrintf ("%s (", AcpiGbl_WordDecode [ACPI_GET_2BIT_FLAG (ResourceType)]
321     AcpiOsPrintf ("%s (", AcpiGbl_WordDecode [ResourceType & 0x3]);

335     /* Decode the general and type-specific flags */

337     if (ResourceType == ACPI_MEMORY_RANGE)
338     {
339         AcpiDmMemoryFlags (Flags, SpecificFlags);
340     }
341     else /* IO range or BusNumberRange */
342     {
343         AcpiDmIoFlags (Flags);
344         if (ResourceType == ACPI_IO_RANGE)
345         {
346             AcpiOsPrintf (" %s,", AcpiGbl_RngDecode [ACPI_GET_2BIT_FLAG (Specifi
334             AcpiOsPrintf (" %s,", AcpiGbl_RngDecode [SpecificFlags & 0x3]);
347         }
348     }
349 }
      unchanged_portion_omitted_


380 /*******************************************************************************
381  *
382  * FUNCTION:    AcpiDmSpaceFlags
383  *
384  * PARAMETERS:  Flags               - Flag byte to be decoded
385  *
386  * RETURN:      None
387  *
388  * DESCRIPTION: Decode the flags specific to Space Address space descriptors
389  *
390  ******************************************************************************/

392 static void
393 AcpiDmSpaceFlags (
394     UINT8                   Flags)
395 {

397     AcpiOsPrintf ("%s, %s, %s, %s,",
398         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Flags)],
399         AcpiGbl_DecDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 1)],
400         AcpiGbl_MinDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 2)],
401         AcpiGbl_MaxDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 3)]);
386         AcpiGbl_ConsumeDecode [(Flags & 1)],
387         AcpiGbl_DecDecode [(Flags & 0x2) >> 1],
388         AcpiGbl_MinDecode [(Flags & 0x4) >> 2],
389         AcpiGbl_MaxDecode [(Flags & 0x8) >> 3]);
402 }


405 /*******************************************************************************
406  *
407  * FUNCTION:    AcpiDmIoFlags
408  *
409  * PARAMETERS:  Flags               - Flag byte to be decoded
410  *
411  * RETURN:      None
412  *
413  * DESCRIPTION: Decode the flags specific to IO Address space descriptors
414  *
415  ******************************************************************************/

417 static void
```

```
418 AcpiDmIoFlags (
419         UINT8               Flags)
420 {
421     AcpiOsPrintf ("%s, %s, %s, %s,",
422         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Flags)],
423         AcpiGbl_MinDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 2)],
424         AcpiGbl_MaxDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 3)],
425         AcpiGbl_DecDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 1)]);
410         AcpiGbl_ConsumeDecode [(Flags & 1)],
411         AcpiGbl_MinDecode [(Flags & 0x4) >> 2],
412         AcpiGbl_MaxDecode [(Flags & 0x8) >> 3],
413         AcpiGbl_DecDecode [(Flags & 0x2) >> 1]);
426 }


429 /*******************************************************************************
430  *
431  * FUNCTION:    AcpiDmIoFlags2
432  *
433  * PARAMETERS:  SpecificFlags       - "Specific" flag byte to be decoded
434  *
435  * RETURN:      None
436  *
437  * DESCRIPTION: Decode the flags specific to IO Address space descriptors
438  *
439  ******************************************************************************/

441 static void
442 AcpiDmIoFlags2 (
443         UINT8               SpecificFlags)
444 {

446     AcpiOsPrintf (", %s",
447         AcpiGbl_TtpDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 4)]);
435         AcpiGbl_TtpDecode [(SpecificFlags & 0x10) >> 4]);

449     /* TRS is only used if TTP is TypeTranslation */

451     if (SpecificFlags & 0x10)
452     {
453         AcpiOsPrintf (", %s",
454             AcpiGbl_TrsDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 5)]);
442             AcpiGbl_TrsDecode [(SpecificFlags & 0x20) >> 5]);
455     }
456 }


459 /*******************************************************************************
460  *
461  * FUNCTION:    AcpiDmMemoryFlags
462  *
463  * PARAMETERS:  Flags               - Flag byte to be decoded
464  *              SpecificFlags       - "Specific" flag byte to be decoded
465  *
466  * RETURN:      None
467  *
468  * DESCRIPTION: Decode flags specific to Memory Address Space descriptors
469  *
470  ******************************************************************************/

472 static void
473 AcpiDmMemoryFlags (
474     UINT8                   Flags,
475     UINT8                   SpecificFlags)
476 {
```



```
478     AcpiOsPrintf ("%s, %s, %s, %s, %s, %s,",
479         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Flags)],
480         AcpiGbl_DecDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 1)],
481         AcpiGbl_MinDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 2)],
482         AcpiGbl_MaxDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 3)],
483         AcpiGbl_MemDecode [ACPI_EXTRACT_2BIT_FLAG (SpecificFlags, 1)],
484         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (SpecificFlags)]);
467         AcpiGbl_ConsumeDecode [(Flags & 1)],
468         AcpiGbl_DecDecode [(Flags & 0x2) >> 1],
469         AcpiGbl_MinDecode [(Flags & 0x4) >> 2],
470         AcpiGbl_MaxDecode [(Flags & 0x8) >> 3],
471         AcpiGbl_MemDecode [(SpecificFlags & 0x6) >> 1],
472         AcpiGbl_RwDecode [(SpecificFlags & 0x1)]);
485 }


488 /*******************************************************************************
489  *
490  * FUNCTION:    AcpiDmMemoryFlags2
491  *
492  * PARAMETERS:  SpecificFlags       - "Specific" flag byte to be decoded
493  *
494  * RETURN:      None
495  *
496  * DESCRIPTION: Decode flags specific to Memory Address Space descriptors
497  *
498  ******************************************************************************/

500 static void
501 AcpiDmMemoryFlags2 (
502     UINT8                   SpecificFlags)
503 {

505     AcpiOsPrintf (", %s, %s",
506         AcpiGbl_MtpDecode [ACPI_EXTRACT_2BIT_FLAG (SpecificFlags, 3)],
507         AcpiGbl_TtpDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 5)]);
494         AcpiGbl_MtpDecode [(SpecificFlags & 0x18) >> 3],
495         AcpiGbl_TtpDecode [(SpecificFlags & 0x20) >> 5]);
508 }


511 /*******************************************************************************
512  *
513  * FUNCTION:    AcpiDmResourceSource
514  *
515  * PARAMETERS:  Resource        - Raw AML descriptor
516  *              MinimumLength   - descriptor length without optional fields
517  *              ResourceLength
518  *
519  * RETURN:      None
520  *
521  * DESCRIPTION: Dump optional ResourceSource fields of an address descriptor
522  *
523  ******************************************************************************/

525 static void
526 AcpiDmResourceSource (
527     AML_RESOURCE            *Resource,
528     ACPI_SIZE               MinimumTotalLength,
529     UINT32                  ResourceLength)
530 {
531     UINT8                   *AmlResourceSource;
532     UINT32                  TotalLength;


535     TotalLength = ResourceLength + sizeof (AML_RESOURCE_LARGE_HEADER);
```

```
537     /* Check if the optional ResourceSource fields are present */

539     if (TotalLength <= MinimumTotalLength)
540     {
541         /* The two optional fields are not used */

543         AcpiOsPrintf (",, ");
544         return;
545     }

547     /* Get a pointer to the ResourceSource */

549     AmlResourceSource = ACPI_ADD_PTR (UINT8, Resource, MinimumTotalLength);

551     /*
552      * Always emit the ResourceSourceIndex (Byte)
553      *
554      * NOTE: Some ASL compilers always create a 0 byte (in the AML) for the
555      * Index even if the String does not exist. Although this is in violation
556      * of the ACPI specification, it is very important to emit ASL code that
557      * can be compiled back to the identical AML. There may be fields and/or
558      * indexes into the resource template buffer that are compiled to absolute
559      * offsets, and these will be broken if the AML length is changed.
560      */
561     AcpiOsPrintf ("0x%2.2X,", (UINT32) AmlResourceSource[0]);

563     /* Make sure that the ResourceSource string exists before dumping it */

565     if (TotalLength > (MinimumTotalLength + 1))
566     {
567         AcpiOsPrintf (" ");
568         AcpiUtPrintString ((char *) &AmlResourceSource[1], ACPI_UINT16_MAX);
556         AcpiUtPrintString ((char *) &AmlResourceSource[1], ACPI_UINT8_MAX);
569     }

571     AcpiOsPrintf (", ");
572 }
```

*unchanged_portion_omitted_*

```
757 /*******************************************************************************
758  *
759  * FUNCTION:    AcpiDmMemory24Descriptor
760  *
761  * PARAMETERS:  Resource            - Pointer to the resource descriptor
762  *              Length              - Length of the descriptor in bytes
763  *              Level               - Current source code indentation level
764  *
765  * RETURN:      None
766  *
767  * DESCRIPTION: Decode a Memory24 descriptor
768  *
769  ******************************************************************************/

771 void
772 AcpiDmMemory24Descriptor (
773     AML_RESOURCE            *Resource,
774     UINT32                  Length,
775     UINT32                  Level)
776 {

778     /* Dump name and read/write flag */

780     AcpiDmIndent (Level);
781     AcpiOsPrintf ("Memory24 (%s,\n",
```



```
782         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (Resource->Memory24.Flags)]);
770         AcpiGbl_RwDecode [Resource->Memory24.Flags & 1]);

784     /* Dump the 4 contiguous WORD values */

786     AcpiDmMemoryFields (&Resource->Memory24.Minimum, 16, Level);

788     /* Insert a descriptor name */

790     AcpiDmIndent (Level + 1);
791     AcpiDmDescriptorName ();
792     AcpiOsPrintf (")\n");
793 }

796 /*******************************************************************************
797  *
798  * FUNCTION:    AcpiDmMemory32Descriptor
799  *
800  * PARAMETERS:  Resource            - Pointer to the resource descriptor
801  *              Length              - Length of the descriptor in bytes
802  *              Level               - Current source code indentation level
803  *
804  * RETURN:      None
805  *
806  * DESCRIPTION: Decode a Memory32 descriptor
807  *
808  ******************************************************************************/

810 void
811 AcpiDmMemory32Descriptor (
812     AML_RESOURCE            *Resource,
813     UINT32                  Length,
814     UINT32                  Level)
815 {

817     /* Dump name and read/write flag */

819     AcpiDmIndent (Level);
820     AcpiOsPrintf ("Memory32 (%s,\n",
821         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (Resource->Memory32.Flags)]);
809         AcpiGbl_RwDecode [Resource->Memory32.Flags & 1]);

823     /* Dump the 4 contiguous DWORD values */

825     AcpiDmMemoryFields (&Resource->Memory32.Minimum, 32, Level);

827     /* Insert a descriptor name */

829     AcpiDmIndent (Level + 1);
830     AcpiDmDescriptorName ();
831     AcpiOsPrintf (")\n");
832 }

835 /*******************************************************************************
836  *
837  * FUNCTION:    AcpiDmFixedMemory32Descriptor
838  *
839  * PARAMETERS:  Resource            - Pointer to the resource descriptor
840  *              Length              - Length of the descriptor in bytes
841  *              Level               - Current source code indentation level
842  *
843  * RETURN:      None
844  *
845  * DESCRIPTION: Decode a Fixed Memory32 descriptor
```

```
 846  *
 847  *****************************************************************************/

 849 void
 850 AcpiDmFixedMemory32Descriptor (
 851     AML_RESOURCE            *Resource,
 852     UINT32                  Length,
 853     UINT32                  Level)
 854 {

 856     /* Dump name and read/write flag */

 858     AcpiDmIndent (Level);
 859     AcpiOsPrintf ("Memory32Fixed (%s,\n",
 860         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (Resource->FixedMemory32.Flags)]);
 848         AcpiGbl_RwDecode [Resource->FixedMemory32.Flags & 1]);

 862     AcpiDmIndent (Level + 1);
 863     AcpiDmDumpInteger32 (Resource->FixedMemory32.Address, "Address Base");

 865     AcpiDmIndent (Level + 1);
 866     AcpiDmDumpInteger32 (Resource->FixedMemory32.AddressLength, "Address Length"

 868     /* Insert a descriptor name */

 870     AcpiDmIndent (Level + 1);
 871     AcpiDmDescriptorName ();
 872     AcpiOsPrintf (")\n");
 873 }
______unchanged_portion_omitted_


 932 /*****************************************************************************
 933  *
 934  * FUNCTION:    AcpiDmInterruptDescriptor
 935  *
 936  * PARAMETERS:  Resource            - Pointer to the resource descriptor
 937  *              Length              - Length of the descriptor in bytes
 938  *              Level               - Current source code indentation level
 939  *
 940  * RETURN:      None
 941  *
 942  * DESCRIPTION: Decode a extended Interrupt descriptor
 943  *
 944  *****************************************************************************/

 946 void
 947 AcpiDmInterruptDescriptor (
 948     AML_RESOURCE            *Resource,
 949     UINT32                  Length,
 950     UINT32                  Level)
 951 {
 952     UINT32                  i;


 955     AcpiDmIndent (Level);
 956     AcpiOsPrintf ("Interrupt (%s, %s, %s, %s, ",
 957         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Resource->ExtendedIrq.Flags)]
 958         AcpiGbl_HeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->ExtendedIrq.Flags, 1
 959         AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->ExtendedIrq.Flags, 2
 960         AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->ExtendedIrq.Flags,
 945         AcpiGbl_ConsumeDecode [(Resource->ExtendedIrq.Flags & 1)],
 946         AcpiGbl_HeDecode [(Resource->ExtendedIrq.Flags >> 1) & 1],
 947         AcpiGbl_LlDecode [(Resource->ExtendedIrq.Flags >> 2) & 1],
 948         AcpiGbl_ShrDecode [(Resource->ExtendedIrq.Flags >> 3) & 1]);
```



```
 962     /*
 963      * The ResourceSource fields are optional and appear after the interrupt
 964      * list. Must compute length based on length of the list. First xrupt
 965      * is included in the struct (reason for -1 below)
 966      */
 967     AcpiDmResourceSource (Resource,
 968         sizeof (AML_RESOURCE_EXTENDED_IRQ) +
 969             ((UINT32) Resource->ExtendedIrq.InterruptCount - 1) * sizeof (UINT32
 970         Resource->ExtendedIrq.ResourceLength);

 972     /* Insert a descriptor name */

 974     AcpiDmDescriptorName ();
 975     AcpiOsPrintf (")\n");

 977     /* Dump the interrupt list */

 979     AcpiDmIndent (Level);
 980     AcpiOsPrintf ("{\n");
 981     for (i = 0; i < Resource->ExtendedIrq.InterruptCount; i++)
 982     {
 983         AcpiDmIndent (Level + 1);
 984         AcpiOsPrintf ("0x%8.8X,\n",
 985             (UINT32) Resource->ExtendedIrq.Interrupts[i]);
 986     }

 988     AcpiDmIndent (Level);
 989     AcpiOsPrintf ("}\n");
 990 }
______unchanged_portion_omitted_

1064 #endif
```

```
**********************************************************
   20667 Thu Dec 26 13:48:47 2013
new/usr/src/common/acpica/components/disassembler/dmresrcl2.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmresrcl2.c - "Large" Resource Descriptor disassembly (#2)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbresrcl2")

/* Local prototypes */

static void
AcpiDmI2cSerialBusDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level);
```



```
static void
AcpiDmSpiSerialBusDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level);

static void
AcpiDmUartSerialBusDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level);

static void
AcpiDmGpioCommon (
    AML_RESOURCE            *Resource,
    UINT32                  Level);

static void
AcpiDmDumpRawDataBuffer (
    UINT8                   *Buffer,
    UINT32                  Length,
    UINT32                  Level);


/* Dispatch table for the serial bus descriptors */

static ACPI_RESOURCE_HANDLER        SerialBusResourceDispatch [] =
{
    NULL,
    AcpiDmI2cSerialBusDescriptor,
    AcpiDmSpiSerialBusDescriptor,
    AcpiDmUartSerialBusDescriptor
};


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDumpRawDataBuffer
 *
 * PARAMETERS:  Buffer              - Pointer to the data bytes
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump a data buffer as a RawDataBuffer() object. Used for
 *              vendor data bytes.
 *
 ******************************************************************************/

static void
AcpiDmDumpRawDataBuffer (
    UINT8                   *Buffer,
    UINT32                  Length,
    UINT32                  Level)
{
    UINT32                  Index;
    UINT32                  i;
    UINT32                  j;


    if (!Length)
    {
        return;
    }
```

```
129     AcpiOsPrintf ("RawDataBuffer (0x%.2X)  // Vendor Data", Length);

131     AcpiOsPrintf ("\n");
132     AcpiDmIndent (Level + 1);
133     AcpiOsPrintf ("{\n");
134     AcpiDmIndent (Level + 2);

136     for (i = 0; i < Length;)
137     {
138         for (j = 0; j < 8; j++)
139         {
140             Index = i + j;
141             if (Index >= Length)
142             {
143                 goto Finish;
144             }

146             AcpiOsPrintf ("0x%2.2X", Buffer[Index]);
147             if ((Index + 1) >= Length)
148             {
149                 goto Finish;
150             }

152             AcpiOsPrintf (", ");
153         }
154         AcpiOsPrintf ("\n");
155         AcpiDmIndent (Level + 2);

157         i += 8;
158     }

160 Finish:
161     AcpiOsPrintf ("\n");
162     AcpiDmIndent (Level + 1);
163     AcpiOsPrintf ("}");
164 }


167 /*******************************************************************************
168  *
169  * FUNCTION:    AcpiDmGpioCommon
170  *
171  * PARAMETERS:  Resource            - Pointer to the resource descriptor
172  *              Level               - Current source code indentation level
173  *
174  * RETURN:      None
175  *
176  * DESCRIPTION: Decode common parts of a GPIO Interrupt descriptor
177  *
178  ******************************************************************************/

180 static void
181 AcpiDmGpioCommon (
182     AML_RESOURCE            *Resource,
183     UINT32                  Level)
184 {
185     UINT32                  PinCount;
186     UINT16                  *PinList;
187     UINT8                   *VendorData;
188     UINT32                  i;


191     /* ResourceSource, ResourceSourceIndex, ResourceType */

193     AcpiDmIndent (Level + 1);
```



```
194     if (Resource->Gpio.ResSourceOffset)
195     {
196         AcpiUtPrintString (
197             ACPI_ADD_PTR (char, Resource, Resource->Gpio.ResSourceOffset),
198             ACPI_UINT16_MAX);
199     }

201     AcpiOsPrintf (", ");
202     AcpiOsPrintf ("0x%2.2X, ", Resource->Gpio.ResSourceIndex);
203     AcpiOsPrintf ("%s, ",
204         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Resource->Gpio.Flags)]);

206     /* Insert a descriptor name */

208     AcpiDmDescriptorName ();
209     AcpiOsPrintf (",");

211     /* Dump the vendor data */

213     if (Resource->Gpio.VendorOffset)
214     {
215         AcpiOsPrintf ("\n");
216         AcpiDmIndent (Level + 1);
217         VendorData = ACPI_ADD_PTR (UINT8, Resource,
218             Resource->Gpio.VendorOffset);

220         AcpiDmDumpRawDataBuffer (VendorData,
221             Resource->Gpio.VendorLength, Level);
222     }

224     AcpiOsPrintf (")\n");

226     /* Dump the interrupt list */

228     AcpiDmIndent (Level + 1);
229     AcpiOsPrintf ("{   // Pin list\n");

231     PinCount = ((UINT32) (Resource->Gpio.ResSourceOffset -
232         Resource->Gpio.PinTableOffset)) /
233         sizeof (UINT16);

235     PinList = (UINT16 *) ACPI_ADD_PTR (char, Resource,
236         Resource->Gpio.PinTableOffset);

238     for (i = 0; i < PinCount; i++)
239     {
240         AcpiDmIndent (Level + 2);
241         AcpiOsPrintf ("0x%4.4X%s\n", PinList[i], ((i + 1) < PinCount) ? "," : ""
242     }

244     AcpiDmIndent (Level + 1);
245     AcpiOsPrintf ("}\n");
246 }


249 /*******************************************************************************
250  *
251  * FUNCTION:    AcpiDmGpioIntDescriptor
252  *
253  * PARAMETERS:  Resource            - Pointer to the resource descriptor
254  *              Length              - Length of the descriptor in bytes
255  *              Level               - Current source code indentation level
256  *
257  * RETURN:      None
258  *
259  * DESCRIPTION: Decode a GPIO Interrupt descriptor
```

```
 *
 *****************************************************************************/

static void
AcpiDmGpioIntDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{

    /* Dump the GpioInt-specific portion of the descriptor */

    /* EdgeLevel, ActiveLevel, Shared */

    AcpiDmIndent (Level);
    AcpiOsPrintf ("GpioInt (%s, %s, %s, ",
        AcpiGbl_HeDecode [ACPI_GET_1BIT_FLAG (Resource->Gpio.IntFlags)],
        AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->Gpio.IntFlags, 1)],
        AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Gpio.IntFlags, 3)])

    /* PinConfig, DebounceTimeout */

    if (Resource->Gpio.PinConfig <= 3)
    {
        AcpiOsPrintf ("%s, ",
            AcpiGbl_PpcDecode[Resource->Gpio.PinConfig]);
    }
    else
    {
        AcpiOsPrintf ("0x%2.2X, ", Resource->Gpio.PinConfig);
    }
    AcpiOsPrintf ("0x%4.4X,\n", Resource->Gpio.DebounceTimeout);

    /* Dump the GpioInt/GpioIo common portion of the descriptor */

    AcpiDmGpioCommon (Resource, Level);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDmGpioIoDescriptor
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a GPIO I/O descriptor
 *
 *****************************************************************************/

static void
AcpiDmGpioIoDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{

    /* Dump the GpioIo-specific portion of the descriptor */

    /* Shared, PinConfig */

    AcpiDmIndent (Level);
    AcpiOsPrintf ("GpioIo (%s, ",
```



```
        AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Gpio.IntFlags, 3)])

    if (Resource->Gpio.PinConfig <= 3)
    {
        AcpiOsPrintf ("%s, ",
            AcpiGbl_PpcDecode[Resource->Gpio.PinConfig]);
    }
    else
    {
        AcpiOsPrintf ("0x%2.2X, ", Resource->Gpio.PinConfig);
    }

    /* DebounceTimeout, DriveStrength, IoRestriction */

    AcpiOsPrintf ("0x%4.4X, ", Resource->Gpio.DebounceTimeout);
    AcpiOsPrintf ("0x%4.4X, ", Resource->Gpio.DriveStrength);
    AcpiOsPrintf ("%s,\n",
        AcpiGbl_IorDecode [ACPI_GET_2BIT_FLAG (Resource->Gpio.IntFlags)]);

    /* Dump the GpioInt/GpioIo common portion of the descriptor */

    AcpiDmGpioCommon (Resource, Level);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDmGpioDescriptor
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a GpioInt/GpioIo GPIO Interrupt/IO descriptor
 *
 *****************************************************************************/

void
AcpiDmGpioDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{
    UINT8                   ConnectionType;


    ConnectionType = Resource->Gpio.ConnectionType;

    switch (ConnectionType)
    {
    case AML_RESOURCE_GPIO_TYPE_INT:

        AcpiDmGpioIntDescriptor (Resource, Length, Level);
        break;

    case AML_RESOURCE_GPIO_TYPE_IO:

        AcpiDmGpioIoDescriptor (Resource, Length, Level);
        break;

    default:

        AcpiOsPrintf ("Unknown GPIO type\n");
        break;
```

```
392     }
393 }


396 /*******************************************************************************
397  *
398  * FUNCTION:    AcpiDmDumpSerialBusVendorData
399  *
400  * PARAMETERS:  Resource            - Pointer to the resource descriptor
401  *
402  * RETURN:      None
403  *
404  * DESCRIPTION: Dump optional serial bus vendor data
405  *
406  ******************************************************************************/

408 static void
409 AcpiDmDumpSerialBusVendorData (
410     AML_RESOURCE            *Resource,
411     UINT32                  Level)
412 {
413     UINT8                   *VendorData;
414     UINT32                  VendorLength;


417     /* Get the (optional) vendor data and length */

419     switch (Resource->CommonSerialBus.Type)
420     {
421     case AML_RESOURCE_I2C_SERIALBUSTYPE:

423         VendorLength = Resource->CommonSerialBus.TypeDataLength -
424             AML_RESOURCE_I2C_MIN_DATA_LEN;

426         VendorData = ACPI_ADD_PTR (UINT8, Resource,
427             sizeof (AML_RESOURCE_I2C_SERIALBUS));
428         break;

430     case AML_RESOURCE_SPI_SERIALBUSTYPE:

432         VendorLength = Resource->CommonSerialBus.TypeDataLength -
433             AML_RESOURCE_SPI_MIN_DATA_LEN;

435         VendorData = ACPI_ADD_PTR (UINT8, Resource,
436             sizeof (AML_RESOURCE_SPI_SERIALBUS));
437         break;

439     case AML_RESOURCE_UART_SERIALBUSTYPE:

441         VendorLength = Resource->CommonSerialBus.TypeDataLength -
442             AML_RESOURCE_UART_MIN_DATA_LEN;

444         VendorData = ACPI_ADD_PTR (UINT8, Resource,
445             sizeof (AML_RESOURCE_UART_SERIALBUS));
446         break;

448     default:

450         return;
451     }

453     /* Dump the vendor bytes as a RawDataBuffer object */

455     AcpiDmDumpRawDataBuffer (VendorData, VendorLength, Level);
456 }
```



```
459 /*******************************************************************************
460  *
461  * FUNCTION:    AcpiDmI2cSerialBusDescriptor
462  *
463  * PARAMETERS:  Resource            - Pointer to the resource descriptor
464  *              Length              - Length of the descriptor in bytes
465  *              Level               - Current source code indentation level
466  *
467  * RETURN:      None
468  *
469  * DESCRIPTION: Decode a I2C serial bus descriptor
470  *
471  ******************************************************************************/

473 static void
474 AcpiDmI2cSerialBusDescriptor (
475     AML_RESOURCE            *Resource,
476     UINT32                  Length,
477     UINT32                  Level)
478 {
479     UINT32                  ResourceSourceOffset;


482     /* SlaveAddress, SlaveMode, ConnectionSpeed, AddressingMode */

484     AcpiDmIndent (Level);
485     AcpiOsPrintf ("I2cSerialBus (0x%4.4X, %s, 0x%8.8X,\n",
486         Resource->I2cSerialBus.SlaveAddress,
487         AcpiGbl_SmDecode [ACPI_GET_1BIT_FLAG (Resource->I2cSerialBus.Flags)],
488         Resource->I2cSerialBus.ConnectionSpeed);

490     AcpiDmIndent (Level + 1);
491     AcpiOsPrintf ("%s, ",
492         AcpiGbl_AmDecode [ACPI_GET_1BIT_FLAG (Resource->I2cSerialBus.TypeSpecifi

494     /* ResourceSource is a required field */

496     ResourceSourceOffset = sizeof (AML_RESOURCE_COMMON_SERIALBUS) +
497         Resource->CommonSerialBus.TypeDataLength;

499     AcpiUtPrintString (
500         ACPI_ADD_PTR (char, Resource, ResourceSourceOffset),
501         ACPI_UINT16_MAX);

503     /* ResourceSourceIndex, ResourceUsage */

505     AcpiOsPrintf (",\n");
506     AcpiDmIndent (Level + 1);
507     AcpiOsPrintf ("0x%2.2X, ", Resource->I2cSerialBus.ResSourceIndex);

509     AcpiOsPrintf ("%s, ",
510         AcpiGbl_ConsumeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->I2cSerialBus.Fl

512     /* Insert a descriptor name */

514     AcpiDmDescriptorName ();
515     AcpiOsPrintf (",\n");

517     /* Dump the vendor data */

519     AcpiDmIndent (Level + 1);
520     AcpiDmDumpSerialBusVendorData (Resource, Level);
521     AcpiOsPrintf (")\n");
522 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiDmSpiSerialBusDescriptor
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a SPI serial bus descriptor
 *
 ******************************************************************************/

static void
AcpiDmSpiSerialBusDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{
    UINT32                  ResourceSourceOffset;


    /* DeviceSelection, DeviceSelectionPolarity, WireMode, DataBitLength */

    AcpiDmIndent (Level);
    AcpiOsPrintf ("SpiSerialBus (0x%4.4X, %s, %s, 0x%2.2X,\n",
        Resource->SpiSerialBus.DeviceSelection,
        AcpiGbl_DpDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->SpiSerialBus.TypeSpe
        AcpiGbl_WmDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.TypeSpecifi
        Resource->SpiSerialBus.DataBitLength);

    /* SlaveMode, ConnectionSpeed, ClockPolarity, ClockPhase */

    AcpiDmIndent (Level + 1);
    AcpiOsPrintf ("%s, 0x%8.8X, %s,\n",
        AcpiGbl_SmDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.Flags)],
        Resource->SpiSerialBus.ConnectionSpeed,
        AcpiGbl_CpoDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.ClockPolar

    AcpiDmIndent (Level + 1);
    AcpiOsPrintf ("%s, ",
        AcpiGbl_CphDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.ClockPhase

    /* ResourceSource is a required field */

    ResourceSourceOffset = sizeof (AML_RESOURCE_COMMON_SERIALBUS) +
        Resource->CommonSerialBus.TypeDataLength;

    AcpiUtPrintString (
        ACPI_ADD_PTR (char, Resource, ResourceSourceOffset),
        ACPI_UINT16_MAX);

    /* ResourceSourceIndex, ResourceUsage */

    AcpiOsPrintf (",\n");
    AcpiDmIndent (Level + 1);
    AcpiOsPrintf ("0x%2.2X, ", Resource->SpiSerialBus.ResSourceIndex);

    AcpiOsPrintf ("%s, ",
        AcpiGbl_ConsumeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->SpiSerialBus.Fl

    /* Insert a descriptor name */

    AcpiDmDescriptorName ();
```



```
    AcpiOsPrintf (",\n");

    /* Dump the vendor data */

    AcpiDmIndent (Level + 1);
    AcpiDmDumpSerialBusVendorData (Resource, Level);
    AcpiOsPrintf (")\n");
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmUartSerialBusDescriptor
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a UART serial bus descriptor
 *
 ******************************************************************************/

static void
AcpiDmUartSerialBusDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{
    UINT32                  ResourceSourceOffset;


    /* ConnectionSpeed, BitsPerByte, StopBits */

    AcpiDmIndent (Level);
    AcpiOsPrintf ("UartSerialBus (0x%8.8X, %s, %s,\n",
        Resource->UartSerialBus.DefaultBaudRate,
        AcpiGbl_BpbDecode [ACPI_EXTRACT_3BIT_FLAG (Resource->UartSerialBus.TypeS
        AcpiGbl_SbDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->UartSerialBus.TypeSp

    /* LinesInUse, IsBigEndian, Parity, FlowControl */

    AcpiDmIndent (Level + 1);
    AcpiOsPrintf ("0x%2.2X, %s, %s, %s,\n",
        Resource->UartSerialBus.LinesEnabled,
        AcpiGbl_EdDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->UartSerialBus.TypeSp
        AcpiGbl_PtDecode [ACPI_GET_3BIT_FLAG (Resource->UartSerialBus.Parity)],
        AcpiGbl_FcDecode [ACPI_GET_2BIT_FLAG (Resource->UartSerialBus.TypeSpecif

    /* ReceiveBufferSize, TransmitBufferSize */

    AcpiDmIndent (Level + 1);
    AcpiOsPrintf ("0x%4.4X, 0x%4.4X, ",
        Resource->UartSerialBus.RxFifoSize,
        Resource->UartSerialBus.TxFifoSize);

    /* ResourceSource is a required field */

    ResourceSourceOffset = sizeof (AML_RESOURCE_COMMON_SERIALBUS) +
        Resource->CommonSerialBus.TypeDataLength;

    AcpiUtPrintString (
        ACPI_ADD_PTR (char, Resource, ResourceSourceOffset),
        ACPI_UINT16_MAX);
```

```
656     /* ResourceSourceIndex, ResourceUsage */

658     AcpiOsPrintf (",\n");
659     AcpiDmIndent (Level + 1);
660     AcpiOsPrintf ("0x%2.2X, ", Resource->UartSerialBus.ResSourceIndex);

662     AcpiOsPrintf ("%s, ",
663         AcpiGbl_ConsumeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->UartSerialBus.F

665     /* Insert a descriptor name */

667     AcpiDmDescriptorName ();
668     AcpiOsPrintf (",\n");

670     /* Dump the vendor data */

672     AcpiDmIndent (Level + 1);
673     AcpiDmDumpSerialBusVendorData (Resource, Level);
674     AcpiOsPrintf (")\n");
675 }


678 /*******************************************************************************
679  *
680  * FUNCTION:    AcpiDmSerialBusDescriptor
681  *
682  * PARAMETERS:  Resource            - Pointer to the resource descriptor
683  *              Length              - Length of the descriptor in bytes
684  *              Level               - Current source code indentation level
685  *
686  * RETURN:      None
687  *
688  * DESCRIPTION: Decode a I2C/SPI/UART serial bus descriptor
689  *
690  ******************************************************************************/

692 void
693 AcpiDmSerialBusDescriptor (
694     AML_RESOURCE            *Resource,
695     UINT32                  Length,
696     UINT32                  Level)
697 {

699     SerialBusResourceDispatch [Resource->CommonSerialBus.Type] (
700         Resource, Length, Level);
701 }

703 #endif
```

```
**********************************************************
   10814 Thu Dec 26 13:48:47 2013
new/usr/src/common/acpica/components/disassembler/dmresrcs.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmresrcs.c - "Small" Resource Descriptor disassembly
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acdisasm.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dbresrcs")


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmIrqDescriptor
 *
```



```
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a IRQ descriptor, either Irq() or IrqNoFlags()
 *
 ******************************************************************************/

void
AcpiDmIrqDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{

    AcpiDmIndent (Level);
    AcpiOsPrintf ("%s (",
        AcpiGbl_IrqDecode [ACPI_GET_1BIT_FLAG (Length)]);
        AcpiGbl_IrqDecode [Length & 1]);

    /* Decode flags byte if present */

    if (Length & 1)
    {
        AcpiOsPrintf ("%s, %s, %s, ",
            AcpiGbl_HeDecode [ACPI_GET_1BIT_FLAG (Resource->Irq.Flags)],
            AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->Irq.Flags, 3)],
            AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Irq.Flags, 4)])
            AcpiGbl_HeDecode [Resource->Irq.Flags & 1],
            AcpiGbl_LlDecode [(Resource->Irq.Flags >> 3) & 1],
            AcpiGbl_ShrDecode [(Resource->Irq.Flags >> 4) & 1]);
    }

    /* Insert a descriptor name */

    AcpiDmDescriptorName ();
    AcpiOsPrintf (")\n");

    AcpiDmIndent (Level + 1);
    AcpiDmBitList (Resource->Irq.IrqMask);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDmaDescriptor
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Length              - Length of the descriptor in bytes
 *              Level               - Current source code indentation level
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode a DMA descriptor
 *
 ******************************************************************************/

void
AcpiDmDmaDescriptor (
    AML_RESOURCE            *Resource,
    UINT32                  Length,
    UINT32                  Level)
{
```

```
122     AcpiDmIndent (Level);
123     AcpiOsPrintf ("DMA (%s, %s, %s, ",
124         AcpiGbl_TypDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Dma.Flags, 5)],
125         AcpiGbl_BmDecode  [ACPI_EXTRACT_1BIT_FLAG (Resource->Dma.Flags, 2)],
126         AcpiGbl_SizDecode [ACPI_GET_2BIT_FLAG (Resource->Dma.Flags)]);
124             AcpiGbl_TypDecode [(Resource->Dma.Flags >> 5) & 3],
125             AcpiGbl_BmDecode  [(Resource->Dma.Flags >> 2) & 1],
126             AcpiGbl_SizDecode [(Resource->Dma.Flags >> 0) & 3]);

128     /* Insert a descriptor name */

130     AcpiDmDescriptorName ();
131     AcpiOsPrintf (")\n");

133     AcpiDmIndent (Level + 1);
134     AcpiDmBitList (Resource->Dma.DmaChannelMask);
135 }


138 /*******************************************************************************
139  *
140  * FUNCTION:    AcpiDmFixedDmaDescriptor
141  *
142  * PARAMETERS:  Resource            - Pointer to the resource descriptor
143  *              Length              - Length of the descriptor in bytes
144  *              Level               - Current source code indentation level
145  *
146  * RETURN:      None
147  *
148  * DESCRIPTION: Decode a FixedDMA descriptor
149  *
150  ******************************************************************************/

152 void
153 AcpiDmFixedDmaDescriptor (
154     AML_RESOURCE            *Resource,
155     UINT32                  Length,
156     UINT32                  Level)
157 {

159     AcpiDmIndent (Level);
160     AcpiOsPrintf ("FixedDMA (0x%4.4X, 0x%4.4X, ",
161         Resource->FixedDma.RequestLines,
162         Resource->FixedDma.Channels);

164     if (Resource->FixedDma.Width <= 5)
165     {
166         AcpiOsPrintf ("%s, ",
167             AcpiGbl_DtsDecode [Resource->FixedDma.Width]);
168     }
169     else
170     {
171         AcpiOsPrintf ("%X /* INVALID DMA WIDTH */, ", Resource->FixedDma.Width);
172     }

174     /* Insert a descriptor name */

176     AcpiDmDescriptorName ();
177     AcpiOsPrintf (")\n");
178 }


181 /*******************************************************************************
182  *
183  * FUNCTION:    AcpiDmIoDescriptor
184  *
```



```
185  * PARAMETERS:  Resource            - Pointer to the resource descriptor
186  *              Length              - Length of the descriptor in bytes
187  *              Level               - Current source code indentation level
188  *
189  * RETURN:      None
190  *
191  * DESCRIPTION: Decode an IO descriptor
192  *
193  ******************************************************************************/

195 void
196 AcpiDmIoDescriptor (
197     AML_RESOURCE            *Resource,
198     UINT32                  Length,
199     UINT32                  Level)
200 {

202     AcpiDmIndent (Level);
203     AcpiOsPrintf ("IO (%s,\n",
204         AcpiGbl_IoDecode [ACPI_GET_1BIT_FLAG (Resource->Io.Flags)]);
161         AcpiGbl_IoDecode [(Resource->Io.Flags & 1)]);

206     AcpiDmIndent (Level + 1);
207     AcpiDmDumpInteger16 (Resource->Io.Minimum, "Range Minimum");

209     AcpiDmIndent (Level + 1);
210     AcpiDmDumpInteger16 (Resource->Io.Maximum, "Range Maximum");

212     AcpiDmIndent (Level + 1);
213     AcpiDmDumpInteger8 (Resource->Io.Alignment, "Alignment");

215     AcpiDmIndent (Level + 1);
216     AcpiDmDumpInteger8 (Resource->Io.AddressLength, "Length");

218     /* Insert a descriptor name */

220     AcpiDmIndent (Level + 1);
221     AcpiDmDescriptorName ();
222     AcpiOsPrintf (")\n");
223 }
```

*unchanged_portion_omitted_*

```
264 /*******************************************************************************
265  *
266  * FUNCTION:    AcpiDmStartDependentDescriptor
267  *
268  * PARAMETERS:  Resource            - Pointer to the resource descriptor
269  *              Length              - Length of the descriptor in bytes
270  *              Level               - Current source code indentation level
271  *
272  * RETURN:      None
273  *
274  * DESCRIPTION: Decode a Start Dependendent functions descriptor
275  *
276  ******************************************************************************/

278 void
279 AcpiDmStartDependentDescriptor (
280     AML_RESOURCE            *Resource,
281     UINT32                  Length,
282     UINT32                  Level)
283 {

285     AcpiDmIndent (Level);
```

```
 287     if (Length & 1)
 288     {
 289         AcpiOsPrintf ("StartDependentFn (0x%2.2X, 0x%2.2X)\n",
 290             (UINT32) ACPI_GET_2BIT_FLAG (Resource->StartDpf.Flags),
 291             (UINT32) ACPI_EXTRACT_2BIT_FLAG (Resource->StartDpf.Flags, 2));
 247             (UINT32) Resource->StartDpf.Flags & 3,
 248             (UINT32) (Resource->StartDpf.Flags >> 2) & 3);
 292     }
 293     else
 294     {
 295         AcpiOsPrintf ("StartDependentFnNoPri ()\n");
 296     }

 298     AcpiDmIndent (Level);
 299     AcpiOsPrintf ("{\n");
 300 }
_____unchanged_portion_omitted_

 357 #endif
```

****
7799 Thu Dec 26 13:48:48 2013
new/usr/src/common/acpica/components/disassembler/dmutils.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dmutils - AML disassembler utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acdisasm.h"

#ifdef ACPI_ASL_COMPILER
#include <acnamesp.h>
#endif

#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmutils")
```



```
/* Data used in keeping track of fields */
#if 0
const char                      *AcpiGbl_FENames[] =
{
    "skip",
    "?access?"
};              /* FE = Field Element */
```

*_____unchanged_portion_omitted_*

```
/******************************************************************************
 *
 * FUNCTION:    AcpiDmDecodeAttribute
 *
 * PARAMETERS:  Attribute       - Attribute field of AccessAs keyword
 *
 * RETURN:      None
 *
 * DESCRIPTION: Decode the AccessAs attribute byte. (Mostly SMBus and
 *              GenericSerialBus stuff.)
 * DESCRIPTION: Decode the AccessAs attribute byte.  (Mostly SMBus stuff)
 *
 *****************************************************************************/

void
AcpiDmDecodeAttribute (
    UINT8                   Attribute)
{

    switch (Attribute)
    {
    case AML_FIELD_ATTRIB_QUICK:
    case AML_FIELD_ATTRIB_SMB_QUICK:

        AcpiOsPrintf ("AttribQuick");
        AcpiOsPrintf ("SMBQuick");
        break;

    case AML_FIELD_ATTRIB_SEND_RCV:
    case AML_FIELD_ATTRIB_SMB_SEND_RCV:

        AcpiOsPrintf ("AttribSendReceive");
        AcpiOsPrintf ("SMBSendReceive");
        break;

    case AML_FIELD_ATTRIB_BYTE:
    case AML_FIELD_ATTRIB_SMB_BYTE:

        AcpiOsPrintf ("AttribByte");
        AcpiOsPrintf ("SMBByte");
        break;

    case AML_FIELD_ATTRIB_WORD:
    case AML_FIELD_ATTRIB_SMB_WORD:

        AcpiOsPrintf ("AttribWord");
        AcpiOsPrintf ("SMBWord");
        break;

    case AML_FIELD_ATTRIB_BLOCK:
    case AML_FIELD_ATTRIB_SMB_WORD_CALL:

        AcpiOsPrintf ("AttribBlock");
        AcpiOsPrintf ("SMBProcessCall");
        break;
```

```
175     case AML_FIELD_ATTRIB_MULTIBYTE:
174     case AML_FIELD_ATTRIB_SMB_BLOCK:

177         AcpiOsPrintf ("AttribBytes");
176         AcpiOsPrintf ("SMBBlock");
178         break;

180     case AML_FIELD_ATTRIB_WORD_CALL:
179     case AML_FIELD_ATTRIB_SMB_BLOCK_CALL:

182         AcpiOsPrintf ("AttribProcessCall");
181         AcpiOsPrintf ("SMBBlockProcessCall");
183         break;

185     case AML_FIELD_ATTRIB_BLOCK_CALL:

187         AcpiOsPrintf ("AttribBlockProcessCall");
188         break;

190     case AML_FIELD_ATTRIB_RAW_BYTES:

192         AcpiOsPrintf ("AttribRawBytes");
193         break;

195     case AML_FIELD_ATTRIB_RAW_PROCESS:

197         AcpiOsPrintf ("AttribRawProcessBytes");
198         break;

200     default:

202         /* A ByteConst is allowed by the grammar */

204         AcpiOsPrintf ("0x%2.2X", Attribute);
186         AcpiOsPrintf ("0x%.2X", Attribute);
205         break;
206     }
207 }
```

*unchanged_portion_omitted_*

```
236 /*****************************************************************************
237  *
238  * FUNCTION:    AcpiDmCommaIfListMember
239  *
240  * PARAMETERS:  Op              - Current operator/operand
241  *
242  * RETURN:      TRUE if a comma was inserted
243  *
244  * DESCRIPTION: Insert a comma if this Op is a member of an argument list.
245  *
246  *****************************************************************************/

248 BOOLEAN
249 AcpiDmCommaIfListMember (
250     ACPI_PARSE_OBJECT       *Op)
251 {

253     if (!Op->Common.Next)
254     {
255         return (FALSE);
237         return FALSE;
256     }

258     if (AcpiDmListType (Op->Common.Parent) & BLOCK_COMMA_LIST)
259     {
```



```
260         /* Check for a NULL target operand */

262         if ((Op->Common.Next->Common.AmlOpcode == AML_INT_NAMEPATH_OP) &&
263             (!Op->Common.Next->Common.Value.String))
264         {
265             /*
266              * To handle the Divide() case where there are two optional
267              * targets, look ahead one more op. If null, this null target
268              * is the one and only target -- no comma needed. Otherwise,
269              * we need a comma to prepare for the next target.
270              */
271             if (!Op->Common.Next->Common.Next)
272             {
273                 return (FALSE);
255                 return FALSE;
274             }
275         }

277         if ((Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST) &&
278             (!(Op->Common.Next->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)))
279         {
280             return (FALSE);
262             return FALSE;
281         }

283         AcpiOsPrintf (", ");
284         return (TRUE);
285     }

287     else if ((Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST) &&
288              (Op->Common.Next->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
289     {
290         AcpiOsPrintf (", ");
291         return (TRUE);
292     }

294     return (FALSE);
295 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   26408 Thu Dec 26 13:48:48 2013
new/usr/src/common/acpica/components/disassembler/dmwalk.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dmwalk - AML disassembly tree walk
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdisasm.h"
#include "acdebug.h"


#ifdef ACPI_DISASSEMBLER

#define _COMPONENT          ACPI_CA_DEBUGGER
        ACPI_MODULE_NAME    ("dmwalk")


#define DB_FULL_OP_INFO     "[%4.4s] @%5.5X #%4.4X:  "
```



```
/* Stub for non-compiler code */

#ifndef ACPI_ASL_COMPILER
void
AcpiDmEmitExternals (
    void)
{
    return;
}
```

*unchanged_portion_omitted_*

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiDmDescendingOp
 *
 * PARAMETERS:  ASL_WALK_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: First visitation of a parse object during tree descent.
 *              Decode opcode name and begin parameter list(s), if any.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiDmDescendingOp (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Level,
    void                    *Context)
{
    ACPI_OP_WALK_INFO       *Info = Context;
    const ACPI_OPCODE_INFO  *OpInfo;
    UINT32                  Name;
    ACPI_PARSE_OBJECT       *NextOp;


    if (Op->Common.DisasmFlags & ACPI_PARSEOP_IGNORE)
    {
        /* Ignore this op -- it was handled elsewhere */

        return (AE_CTRL_DEPTH);
    }

    /* Level 0 is at the Definition Block level */

    if (Level == 0)
    {
        /* In verbose mode, print the AML offset, opcode and depth count */

        if (Info->WalkState)
        {
            VERBOSE_PRINT ((DB_FULL_OP_INFO,
                (Info->WalkState->MethodNode ?
                    Info->WalkState->MethodNode->Name.Ascii : "   "),
                Op->Common.AmlOffset, (UINT32) Op->Common.AmlOpcode));
        }

        if (Op->Common.AmlOpcode == AML_SCOPE_OP)
        {
            /* This is the beginning of the Definition Block */

            AcpiOsPrintf ("{\n");

            /* Emit all External() declarations here */
```

```
            AcpiDmEmitExternals ();
            return (AE_OK);
        }
    }
    else if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
             (!(Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)) &&
             (Op->Common.AmlOpcode != AML_INT_BYTELIST_OP))
    {
            /*
             * This is a first-level element of a term list,
             * indent a new line
             */
            switch (Op->Common.AmlOpcode)
            {
            case AML_NOOP_OP:
                /*
                 * Optionally just ignore this opcode. Some tables use
                 * NoOp opcodes for "padding" out packages that the BIOS
                 * changes dynamically. This can leave hundreds or
                 * thousands of NoOp opcodes that if disassembled,
                 * cannot be compiled because they are syntactically
                 * incorrect.
                 */
                if (AcpiGbl_IgnoreNoopOperator)
                {
                    Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
                    return (AE_OK);
                }

                /* Fallthrough */

            default:

                AcpiDmIndent (Level);
                break;
            }

            Info->LastLevel = Level;
            Info->Count = 0;
    }

    /*
     * This is an inexpensive mechanism to try and keep lines from getting
     * too long. When the limit is hit, start a new line at the previous
     * indent plus one. A better but more expensive mechanism would be to
     * keep track of the current column.
     */
    Info->Count++;
    if (Info->Count /* +Info->LastLevel */ > 10)
    if (Info->Count /*+Info->LastLevel*/ > 10)
    {
        Info->Count = 0;
        AcpiOsPrintf ("\n");
        AcpiDmIndent (Info->LastLevel + 1);
    }

    /* Print the opcode name */

    AcpiDmDisassembleOneOp (NULL, Info, Op);

    if ((Op->Common.DisasmOpcode == ACPI_DASM_LNOT_PREFIX) ||
        (Op->Common.AmlOpcode == AML_INT_CONNECTION_OP))
    if (Op->Common.DisasmOpcode == ACPI_DASM_LNOT_PREFIX)
    {
        return (AE_OK);
```



```
    }

    if ((Op->Common.AmlOpcode == AML_NAME_OP) ||
        (Op->Common.AmlOpcode == AML_RETURN_OP))
    {
        Info->Level--;
    }

    /* Start the opcode argument list if necessary */

    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

    if ((OpInfo->Flags & AML_HAS_ARGS) ||
        (Op->Common.AmlOpcode == AML_EVENT_OP))
    {
        /* This opcode has an argument list */

        if (AcpiDmBlockType (Op) & BLOCK_PAREN)
        {
            AcpiOsPrintf (" (");
        }

        /* If this is a named opcode, print the associated name value */

        if (OpInfo->Flags & AML_NAMED)
        {
            switch (Op->Common.AmlOpcode)
            {
            case AML_ALIAS_OP:

                NextOp = AcpiPsGetDepthNext (NULL, Op);
                NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
                AcpiDmNamestring (NextOp->Common.Value.Name);
                AcpiOsPrintf (", ");

                /*lint -fallthrough */

            default:

                Name = AcpiPsGetName (Op);
                if (Op->Named.Path)
                {
                    AcpiDmNamestring ((char *) Op->Named.Path);
                }
                else
                {
                    AcpiDmDumpName (Name);
                }

                if (Op->Common.AmlOpcode != AML_INT_NAMEDFIELD_OP)
                {
                    if (AcpiGbl_DbOpt_verbose)
                    {
                        (void) AcpiPsDisplayObjectPathname (NULL, Op);
                    }
                }
                break;
            }

            switch (Op->Common.AmlOpcode)
            {
            case AML_METHOD_OP:

                AcpiDmMethodFlags (Op);
                AcpiOsPrintf (")");
```

```
561                 /* Emit description comment for Method() with a predefined ACPI
563                 AcpiDmPredefinedDescription (Op);
564                 break;


567             case AML_NAME_OP:

569                 /* Check for _HID and related EISAID() */

571                 AcpiDmIsEisaId (Op);
572                 AcpiOsPrintf (", ");
573                 break;


576             case AML_REGION_OP:

578                 AcpiDmRegionFlags (Op);
579                 break;


582             case AML_POWER_RES_OP:

584                 /* Mark the next two Ops as part of the parameter list */

586                 AcpiOsPrintf (", ");
587                 NextOp = AcpiPsGetDepthNext (NULL, Op);
588                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;

590                 NextOp = NextOp->Common.Next;
591                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;
592                 return (AE_OK);


595             case AML_PROCESSOR_OP:

597                 /* Mark the next three Ops as part of the parameter list */

599                 AcpiOsPrintf (", ");
600                 NextOp = AcpiPsGetDepthNext (NULL, Op);
601                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;

603                 NextOp = NextOp->Common.Next;
604                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;

606                 NextOp = NextOp->Common.Next;
607                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;
608                 return (AE_OK);


611             case AML_MUTEX_OP:
612             case AML_DATA_REGION_OP:

614                 AcpiOsPrintf (", ");
615                 return (AE_OK);


618             case AML_EVENT_OP:
619             case AML_ALIAS_OP:

621                 return (AE_OK);


624             case AML_SCOPE_OP:
625             case AML_DEVICE_OP:
626             case AML_THERMAL_ZONE_OP:
```



```
628                 AcpiOsPrintf (")");
629                 break;


632             default:

634                 AcpiOsPrintf ("*** Unhandled named opcode %X\n",
635                     Op->Common.AmlOpcode);
605                 AcpiOsPrintf ("*** Unhandled named opcode %X\n", Op->Common.AmlO
636                 break;
637             }
638         }

640         else switch (Op->Common.AmlOpcode)
641         {
642         case AML_FIELD_OP:
643         case AML_BANK_FIELD_OP:
644         case AML_INDEX_FIELD_OP:

646             Info->BitOffset = 0;

648             /* Name of the parent OperationRegion */

650             NextOp = AcpiPsGetDepthNext (NULL, Op);
651             AcpiDmNamestring (NextOp->Common.Value.Name);
652             AcpiOsPrintf (", ");
653             NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;

655             switch (Op->Common.AmlOpcode)
656             {
657             case AML_BANK_FIELD_OP:

659                 /* Namestring - Bank Name */

661                 NextOp = AcpiPsGetDepthNext (NULL, NextOp);
662                 AcpiDmNamestring (NextOp->Common.Value.Name);
663                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
664                 AcpiOsPrintf (", ");

666                 /*
667                  * Bank Value. This is a TermArg in the middle of the parameter
668                  * list, must handle it here.
669                  *
670                  * Disassemble the TermArg parse tree. ACPI_PARSEOP_PARAMLIST
671                  * eliminates newline in the output.
672                  */
673                 NextOp = NextOp->Common.Next;

675                 Info->Flags = ACPI_PARSEOP_PARAMLIST;
676                 AcpiDmWalkParseTree (NextOp, AcpiDmDescendingOp,
677                     AcpiDmAscendingOp, Info);
646                 AcpiDmWalkParseTree (NextOp, AcpiDmDescendingOp, AcpiDmAscending
678                 Info->Flags = 0;
679                 Info->Level = Level;

681                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
682                 AcpiOsPrintf (", ");
683                 break;

685             case AML_INDEX_FIELD_OP:

687                 /* Namestring - Data Name */

689                 NextOp = AcpiPsGetDepthNext (NULL, NextOp);
690                 AcpiDmNamestring (NextOp->Common.Value.Name);
```

```
691                 AcpiOsPrintf (", ");
692                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
693                 break;

695             default:

697                 break;
698             }

700             AcpiDmFieldFlags (NextOp);
701             break;


703         case AML_BUFFER_OP:

705             /* The next op is the size parameter */

707             NextOp = AcpiPsGetDepthNext (NULL, Op);
708             if (!NextOp)
709             {
710                 /* Single-step support */

712                 return (AE_OK);
713             }

715             if (Op->Common.DisasmOpcode == ACPI_DASM_RESOURCE)
716             {
717                 /*
718                  * We have a resource list. Don't need to output
719                  * the buffer size Op. Open up a new block
720                  */
721                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
722                 NextOp = NextOp->Common.Next;
723                 AcpiOsPrintf (")");

725                 /* Emit description comment for Name() with a predefined ACPI na

727                 AcpiDmPredefinedDescription (Op->Asl.Parent);

729                 AcpiOsPrintf ("\n");
693                 AcpiOsPrintf (")\n");
730                 AcpiDmIndent (Info->Level);
731                 AcpiOsPrintf ("{\n");
732                 return (AE_OK);
733             }

735             /* Normal Buffer, mark size as in the parameter list */

737             NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;
738             return (AE_OK);


740         case AML_VAR_PACKAGE_OP:
741         case AML_IF_OP:
742         case AML_WHILE_OP:

744             /* The next op is the size or predicate parameter */

746             NextOp = AcpiPsGetDepthNext (NULL, Op);
747             if (NextOp)
748             {
749                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;
750             }
751             return (AE_OK);
```



```
753         case AML_PACKAGE_OP:

755             /* The next op is the size parameter */
721             /* The next op is the size or predicate parameter */

757             NextOp = AcpiPsGetDepthNext (NULL, Op);
758             if (NextOp)
759             {
760                 NextOp->Common.DisasmFlags |= ACPI_PARSEOP_PARAMLIST;
761             }
762             return (AE_OK);


764         case AML_MATCH_OP:

766             AcpiDmMatchOp (Op);
767             break;


769         default:

771             break;
772         }

774         if (AcpiDmBlockType (Op) & BLOCK_BRACE)
775         {
776             AcpiOsPrintf ("\n");
777             AcpiDmIndent (Level);
778             AcpiOsPrintf ("{\n");
779         }
780     }

782     return (AE_OK);
783 }


786 /*******************************************************************************
787  *
788  * FUNCTION:    AcpiDmAscendingOp
789  *
790  * PARAMETERS:  ASL_WALK_CALLBACK
791  *
792  * RETURN:      Status
793  *
794  * DESCRIPTION: Second visitation of a parse object, during ascent of parse
795  *              tree. Close out any parameter lists and complete the opcode.
796  *
797  ******************************************************************************/

799 static ACPI_STATUS
800 AcpiDmAscendingOp (
801     ACPI_PARSE_OBJECT       *Op,
802     UINT32                  Level,
803     void                    *Context)
804 {
805     ACPI_OP_WALK_INFO       *Info = Context;
806     ACPI_PARSE_OBJECT       *ParentOp;


809     if (Op->Common.DisasmFlags & ACPI_PARSEOP_IGNORE)
810     {
811         /* Ignore this op -- it was handled elsewhere */

813         return (AE_OK);
814     }
```

```
816     if ((Level == 0) && (Op->Common.AmlOpcode == AML_SCOPE_OP))
817     {
818         /* Indicates the end of the current descriptor block (table) */

820         AcpiOsPrintf ("}\n\n");
821         return (AE_OK);
822     }

824     switch (AcpiDmBlockType (Op))
825     {
826     case BLOCK_PAREN:

828         /* Completed an op that has arguments, add closing paren */

830         AcpiOsPrintf (")");

832         if (Op->Common.AmlOpcode == AML_NAME_OP)
833         {
834             /* Emit description comment for Name() with a predefined ACPI name *

836             AcpiDmPredefinedDescription (Op);
837         }
838         else
839         {
840             /* For Create* operators, attempt to emit resource tag description *

842             AcpiDmFieldPredefinedDescription (Op);
843         }

845         /* Could be a nested operator, check if comma required */

847         if (!AcpiDmCommaIfListMember (Op))
848         {
849             if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
850                     (!(Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)) &&
851                     (Op->Common.AmlOpcode != AML_INT_BYTELIST_OP))
852             {
853                 /*
854                  * This is a first-level element of a term list
855                  * start a new line
856                  */
857                 if (!(Info->Flags & ACPI_PARSEOP_PARAMLIST))
858                 {
859                     AcpiOsPrintf ("\n");
860                 }
861             }
862         }
863         break;


865     case BLOCK_BRACE:
866     case (BLOCK_BRACE | BLOCK_PAREN):

868         /* Completed an op that has a term list, add closing brace */

870         if (Op->Common.DisasmFlags & ACPI_PARSEOP_EMPTY_TERMLIST)
871         {
872             AcpiOsPrintf ("}");
873         }
874         else
875         {
876             AcpiDmIndent (Level);
877             AcpiOsPrintf ("}");
878         }

880         AcpiDmCommaIfListMember (Op);
```



```
882         if (AcpiDmBlockType (Op->Common.Parent) != BLOCK_PAREN)
883         {
884             AcpiOsPrintf ("\n");
885             if (!(Op->Common.DisasmFlags & ACPI_PARSEOP_EMPTY_TERMLIST))
886             {
887                 if ((Op->Common.AmlOpcode == AML_IF_OP)  &&
888                     (Op->Common.Next) &&
889                     (Op->Common.Next->Common.AmlOpcode == AML_ELSE_OP))
890                 {
891                     break;
892                 }

894                 if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
895                     (!Op->Common.Next))
896                 {
897                     break;
898                 }
899                 AcpiOsPrintf ("\n");
900             }
901         }
902         break;


904     case BLOCK_NONE:
905     default:

907         /* Could be a nested operator, check if comma required */

909         if (!AcpiDmCommaIfListMember (Op))
910         {
911             if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
912                     (!(Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)) &&
913                     (Op->Common.AmlOpcode != AML_INT_BYTELIST_OP))
914             {
915                 /*
916                  * This is a first-level element of a term list
917                  * start a new line
918                  */
919                 AcpiOsPrintf ("\n");
920             }
921         }
922         else if (Op->Common.Parent)
923         {
924             switch (Op->Common.Parent->Common.AmlOpcode)
925             {
926             case AML_PACKAGE_OP:
927             case AML_VAR_PACKAGE_OP:

929                 if (!(Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
930                 {
931                     AcpiOsPrintf ("\n");
932                 }
933                 break;

935             default:

937                 break;
938             }
939         }
940         break;
941     }

943     if (Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)
944     {
945         if ((Op->Common.Next) &&
```

```
946             (Op->Common.Next->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
947         {
948             return (AE_OK);
949         }
950
951         /*
952          * Just completed a parameter node for something like "Buffer (param)".
953          * Close the paren and open up the term list block with a brace
954          */
955         if (Op->Common.Next)
956         {
957             AcpiOsPrintf (")");
958
959             /* Emit description comment for Name() with a predefined ACPI name */
960
961             ParentOp = Op->Common.Parent;
962             if (ParentOp)
963             {
964                 ParentOp = ParentOp->Common.Parent;
965                 if (ParentOp && ParentOp->Asl.AmlOpcode == AML_NAME_OP)
966                 {
967                     AcpiDmPredefinedDescription (ParentOp);
968                 }
969             }
970             AcpiOsPrintf ("\n");
913             AcpiOsPrintf (")\n");
971             AcpiDmIndent (Level - 1);
972             AcpiOsPrintf ("{\n");
973         }
974         else
975         {
976             Op->Common.Parent->Common.DisasmFlags |=
977                                     ACPI_PARSEOP_EMPTY_TERMLIST;
978             AcpiOsPrintf (") {");
979         }
980     }
981
982     if ((Op->Common.AmlOpcode == AML_NAME_OP) ||
983         (Op->Common.AmlOpcode == AML_RETURN_OP))
984     {
985         Info->Level++;
986     }
987     return (AE_OK);
988 }
```

*unchanged_portion_omitted_*

****
12480 Thu Dec 26 13:48:49 2013
new/usr/src/common/acpica/components/dispatcher/dsargs.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsargs - Support for execution of dynamic arguments for static
   4  *                       objects (regions, fields, buffer fields, etc.)
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  45 #define __DSARGS_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acparser.h"
  50 #include "amlcode.h"
  51 #include "acdispat.h"
  52 #include "acnamesp.h"

  54 #define _COMPONENT          ACPI_DISPATCHER
  55         ACPI_MODULE_NAME    ("dsargs")

  57 /* Local prototypes */

  59 static ACPI_STATUS
```



```
  60 AcpiDsExecuteArguments (
  61     ACPI_NAMESPACE_NODE     *Node,
  62     ACPI_NAMESPACE_NODE     *ScopeNode,
  63     UINT32                  AmlLength,
  64     UINT8                   *AmlStart);


  67 /******************************************************************************
  68  *
  69  * FUNCTION:    AcpiDsExecuteArguments
  70  *
  71  * PARAMETERS:  Node                - Object NS node
  72  *              ScopeNode           - Parent NS node
  73  *              AmlLength           - Length of executable AML
  74  *              AmlStart            - Pointer to the AML
  75  *
  76  * RETURN:      Status.
  77  *
  78  * DESCRIPTION: Late (deferred) execution of region or field arguments
  79  *
  80  *****************************************************************************/

  82 static ACPI_STATUS
  83 AcpiDsExecuteArguments (
  84     ACPI_NAMESPACE_NODE     *Node,
  85     ACPI_NAMESPACE_NODE     *ScopeNode,
  86     UINT32                  AmlLength,
  87     UINT8                   *AmlStart)
  88 {
  89     ACPI_STATUS             Status;
  90     ACPI_PARSE_OBJECT       *Op;
  91     ACPI_WALK_STATE         *WalkState;


  94     ACPI_FUNCTION_TRACE (DsExecuteArguments);


  97     /* Allocate a new parser op to be the root of the parsed tree */

  99     Op = AcpiPsAllocOp (AML_INT_EVAL_SUBTREE_OP);
 100     if (!Op)
 101     {
 102         return_ACPI_STATUS (AE_NO_MEMORY);
 103     }

 105     /* Save the Node for use in AcpiPsParseAml */

 107     Op->Common.Node = ScopeNode;

 109     /* Create and initialize a new parser state */

 111     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
 112     if (!WalkState)
 113     {
 114         Status = AE_NO_MEMORY;
 115         goto Cleanup;
 116     }

 118     Status = AcpiDsInitAmlWalk (WalkState, Op, NULL, AmlStart,
 119                 AmlLength, NULL, ACPI_IMODE_LOAD_PASS1);
 120     if (ACPI_FAILURE (Status))
 121     {
 122         AcpiDsDeleteWalkState (WalkState);
 123         goto Cleanup;
 124     }
```

```
126     /* Mark this parse as a deferred opcode */

128     WalkState->ParseFlags = ACPI_PARSE_DEFERRED_OP;
129     WalkState->DeferredNode = Node;

131     /* Pass1: Parse the entire declaration */

133     Status = AcpiPsParseAml (WalkState);
134     if (ACPI_FAILURE (Status))
135     {
136         goto Cleanup;
137     }

139     /* Get and init the Op created above */

141     Op->Common.Node = Node;
142     AcpiPsDeleteParseTree (Op);

144     /* Evaluate the deferred arguments */

146     Op = AcpiPsAllocOp (AML_INT_EVAL_SUBTREE_OP);
147     if (!Op)
148     {
149         return_ACPI_STATUS (AE_NO_MEMORY);
150     }

152     Op->Common.Node = ScopeNode;

154     /* Create and initialize a new parser state */

156     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
157     if (!WalkState)
158     {
159         Status = AE_NO_MEMORY;
160         goto Cleanup;
161     }

163     /* Execute the opcode and arguments */

165     Status = AcpiDsInitAmlWalk (WalkState, Op, NULL, AmlStart,
166                     AmlLength, NULL, ACPI_IMODE_EXECUTE);
167     if (ACPI_FAILURE (Status))
168     {
169         AcpiDsDeleteWalkState (WalkState);
170         goto Cleanup;
171     }

173     /* Mark this execution as a deferred opcode */

175     WalkState->DeferredNode = Node;
176     Status = AcpiPsParseAml (WalkState);

178 Cleanup:
179     AcpiPsDeleteParseTree (Op);
180     return_ACPI_STATUS (Status);
181 }
```

*unchanged_portion_omitted_*

```
380 /*****************************************************************************
381  *
382  * FUNCTION:    AcpiDsGetRegionArguments
383  *
384  * PARAMETERS:  ObjDesc         - A valid region object
385  *
386  * RETURN:      Status.
```



```
387  *
388  * DESCRIPTION: Get region address and length. This implements the late
389  *              evaluation of these region attributes.
390  *
391  *****************************************************************************/

393 ACPI_STATUS
394 AcpiDsGetRegionArguments (
395     ACPI_OPERAND_OBJECT     *ObjDesc)
396 {
397     ACPI_NAMESPACE_NODE     *Node;
398     ACPI_STATUS             Status;
399     ACPI_OPERAND_OBJECT     *ExtraDesc;


402     ACPI_FUNCTION_TRACE_PTR (DsGetRegionArguments, ObjDesc);


405     if (ObjDesc->Region.Flags & AOPOBJ_DATA_VALID)
406     {
407         return_ACPI_STATUS (AE_OK);
408     }

410     ExtraDesc = AcpiNsGetSecondaryObject (ObjDesc);
411     if (!ExtraDesc)
412     {
413         return_ACPI_STATUS (AE_NOT_EXIST);
414     }

416     /* Get the Region node */

418     Node = ObjDesc->Region.Node;

420     ACPI_DEBUG_EXEC (AcpiUtDisplayInitPathname (ACPI_TYPE_REGION, Node, NULL));

422     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "[%4.4s] OpRegion Arg Init at AML %p\n",
423         AcpiUtGetNodeName (Node), ExtraDesc->Extra.AmlStart));

425     /* Execute the argument AML */

427     Status = AcpiDsExecuteArguments (Node, ExtraDesc->Extra.ScopeNode,
427     Status = AcpiDsExecuteArguments (Node, Node->Parent,
428                 ExtraDesc->Extra.AmlLength, ExtraDesc->Extra.AmlStart);
429     if (ACPI_FAILURE (Status))
430     {
431         return_ACPI_STATUS (Status);
432     }

434     Status = AcpiUtAddAddressRange (ObjDesc->Region.SpaceId,
435                 ObjDesc->Region.Address, ObjDesc->Region.Length,
436                 Node);
437     return_ACPI_STATUS (Status);
438 }
```

*unchanged_portion_omitted_*

****
12954 Thu Dec 26 13:48:49 2013
new/usr/src/common/acpica/components/dispatcher/dscontrol.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dscontrol - Support for execution control opcodes -
   4  *                          if/else/while/return
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  45 #define __DSCONTROL_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "amlcode.h"
  50 #include "acdispat.h"
  51 #include "acinterp.h"

  53 #define _COMPONENT          ACPI_DISPATCHER
  54         ACPI_MODULE_NAME    ("dscontrol")


  57 /*****************************************************************************
  58  *
  59  * FUNCTION:    AcpiDsExecBeginControlOp
```



```
  60  *
  61  * PARAMETERS:  WalkList        - The list that owns the walk stack
  62  *              Op              - The control Op
  63  *
  64  * RETURN:      Status
  65  *
  66  * DESCRIPTION: Handles all control ops encountered during control method
  67  *              execution.
  68  *
  69  *****************************************************************************/

  71 ACPI_STATUS
  72 AcpiDsExecBeginControlOp (
  73     ACPI_WALK_STATE         *WalkState,
  74     ACPI_PARSE_OBJECT       *Op)
  75 {
  76     ACPI_STATUS             Status = AE_OK;
  77     ACPI_GENERIC_STATE      *ControlState;


  80     ACPI_FUNCTION_NAME (DsExecBeginControlOp);


  83     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p Opcode=%2.2X State=%p\n",
  84         Op, Op->Common.AmlOpcode, WalkState));

  86     switch (Op->Common.AmlOpcode)
  87     {
  88     case AML_WHILE_OP:

  89         /*
  90          * If this is an additional iteration of a while loop, continue.
  91          * There is no need to allocate a new control state.
  92          */
  93         if (WalkState->ControlState)
  94         {
  95             if (WalkState->ControlState->Control.AmlPredicateStart ==
  96                 (WalkState->ParserState.Aml - 1))
  97             {
  98                 /* Reset the state to start-of-loop */

 100                 WalkState->ControlState->Common.State =
 101                     ACPI_CONTROL_CONDITIONAL_EXECUTING;
 102                 break;
 103             }
 104         }

 106         /*lint -fallthrough */

 108     case AML_IF_OP:

 109         /*
 110          * IF/WHILE: Create a new control state to manage these
 111          * constructs. We need to manage these as a stack, in order
 112          * to handle nesting.
 113          */
 114         ControlState = AcpiUtCreateControlState ();
 115         if (!ControlState)
 116         {
 117             Status = AE_NO_MEMORY;
 118             break;
 119         }
 120         /*
 121          * Save a pointer to the predicate for multiple executions
 122          * of a loop
 123          */
```

```
124         ControlState->Control.AmlPredicateStart = WalkState->ParserState.Aml - 1
125         ControlState->Control.PackageEnd = WalkState->ParserState.PkgEnd;
126         ControlState->Control.Opcode = Op->Common.AmlOpcode;


129         /* Push the control state on this walk's control stack */

131         AcpiUtPushGenericState (&WalkState->ControlState, ControlState);
132         break;

134     case AML_ELSE_OP:

136         /* Predicate is in the state object */
137         /* If predicate is true, the IF was executed, ignore ELSE part */

139         if (WalkState->LastPredicate)
140         {
141             Status = AE_CTRL_TRUE;
142         }

144         break;

146     case AML_RETURN_OP:

148         break;

150     default:

152         break;
153     }

155     return (Status);
156 }


159 /*****************************************************************************
160  *
161  * FUNCTION:    AcpiDsExecEndControlOp
162  *
163  * PARAMETERS:  WalkList        - The list that owns the walk stack
164  *              Op              - The control Op
165  *
166  * RETURN:      Status
167  *
168  * DESCRIPTION: Handles all control ops encountered during control method
169  *              execution.
170  *
171  ****************************************************************************/

173 ACPI_STATUS
174 AcpiDsExecEndControlOp (
175     ACPI_WALK_STATE         *WalkState,
176     ACPI_PARSE_OBJECT       *Op)
177 {
178     ACPI_STATUS             Status = AE_OK;
179     ACPI_GENERIC_STATE      *ControlState;


182     ACPI_FUNCTION_NAME (DsExecEndControlOp);


185     switch (Op->Common.AmlOpcode)
186     {
187     case AML_IF_OP:

189         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "[IF_OP] Op=%p\n", Op));
```



```
191         /*
192          * Save the result of the predicate in case there is an
193          * ELSE to come
194          */
195         WalkState->LastPredicate =
196             (BOOLEAN) WalkState->ControlState->Common.Value;

198         /*
199          * Pop the control state that was created at the start
200          * of the IF and free it
201          */
202         ControlState = AcpiUtPopGenericState (&WalkState->ControlState);
203         AcpiUtDeleteGenericState (ControlState);
204         break;


206     case AML_ELSE_OP:

208         break;


210     case AML_WHILE_OP:

212         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "[WHILE_OP] Op=%p\n", Op));

214         ControlState = WalkState->ControlState;
215         if (ControlState->Common.Value)
216         {
217             /* Predicate was true, the body of the loop was just executed */

219             /*
220              * This loop counter mechanism allows the interpreter to escape
221              * possibly infinite loops. This can occur in poorly written AML
222              * when the hardware does not respond within a while loop and the
223              * loop does not implement a timeout.
224              */
225             ControlState->Control.LoopCount++;
226             if (ControlState->Control.LoopCount > ACPI_MAX_LOOP_ITERATIONS)
227             {
228                 Status = AE_AML_INFINITE_LOOP;
229                 break;
230             }

232             /*
233              * Go back and evaluate the predicate and maybe execute the loop
234              * another time
235              */
236             Status = AE_CTRL_PENDING;
237             WalkState->AmlLastWhile = ControlState->Control.AmlPredicateStart;
238             break;
239         }

241         /* Predicate was false, terminate this while loop */

243         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
244             "[WHILE_OP] termination! Op=%p\n",Op));

246         /* Pop this control state and free it */

248         ControlState = AcpiUtPopGenericState (&WalkState->ControlState);
249         AcpiUtDeleteGenericState (ControlState);
250         break;


252     case AML_RETURN_OP:
```

```
254         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
255             "[RETURN_OP] Op=%p Arg=%p\n",Op, Op->Common.Value.Arg));
257         /*
258          * One optional operand -- the return value
259          * It can be either an immediate operand or a result that
260          * has been bubbled up the tree
261          */
262         if (Op->Common.Value.Arg)
263         {
264             /* Since we have a real Return(), delete any implicit return */
266             AcpiDsClearImplicitReturn (WalkState);
268             /* Return statement has an immediate operand */
270             Status = AcpiDsCreateOperands (WalkState, Op->Common.Value.Arg);
271             if (ACPI_FAILURE (Status))
272             {
273                 return (Status);
274             }
276             /*
277              * If value being returned is a Reference (such as
278              * an arg or local), resolve it now because it may
279              * cease to exist at the end of the method.
280              */
281             Status = AcpiExResolveToValue (&WalkState->Operands [0], WalkState);
282             if (ACPI_FAILURE (Status))
283             {
284                 return (Status);
285             }
287             /*
288              * Get the return value and save as the last result
289              * value. This is the only place where WalkState->ReturnDesc
290              * is set to anything other than zero!
291              */
292             WalkState->ReturnDesc = WalkState->Operands[0];
293         }
294         else if (WalkState->ResultCount)
295         {
296             /* Since we have a real Return(), delete any implicit return */
298             AcpiDsClearImplicitReturn (WalkState);
300             /*
301              * The return value has come from a previous calculation.
302              *
303              * If value being returned is a Reference (such as
304              * an arg or local), resolve it now because it may
305              * cease to exist at the end of the method.
306              *
307              * Allow references created by the Index operator to return
308              * unchanged.
309              */
310             if ((ACPI_GET_DESCRIPTOR_TYPE (WalkState->Results->Results.ObjDesc[0
311                 ((WalkState->Results->Results.ObjDesc [0])->Common.Type == ACPI_
312                 ((WalkState->Results->Results.ObjDesc [0])->Reference.Class != A
313             {
314                 Status = AcpiExResolveToValue (&WalkState->Results->Results.ObjD
315                 if (ACPI_FAILURE (Status))
316                 {
317                     return (Status);
318                 }
```



```
319             }
321             WalkState->ReturnDesc = WalkState->Results->Results.ObjDesc [0];
322         }
323         else
324         {
325             /* No return operand */
327             if (WalkState->NumOperands)
328             {
329                 AcpiUtRemoveReference (WalkState->Operands [0]);
330             }
332             WalkState->Operands [0]     = NULL;
333             WalkState->NumOperands      = 0;
334             WalkState->ReturnDesc       = NULL;
335         }
338         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
339             "Completed RETURN_OP State=%p, RetVal=%p\n",
340             WalkState, WalkState->ReturnDesc));
342         /* End the control method execution right now */
344         Status = AE_CTRL_TERMINATE;
345         break;
347     case AML_NOOP_OP:
349         /* Just do nothing! */
351         break;
353     case AML_BREAK_POINT_OP:
355         /*
356          * Set the single-step flag. This will cause the debugger (if present)
357          * to break to the console within the AML debugger at the start of the
358          * next AML instruction.
359          */
360         ACPI_DEBUGGER_EXEC (
361             AcpiGbl_CmSingleStep = TRUE);
362         ACPI_DEBUGGER_EXEC (
363             AcpiOsPrintf ("**break** Executed AML BreakPoint opcode\n"));
365         /* Call to the OSL in case OS wants a piece of the action */
367         Status = AcpiOsSignal (ACPI_SIGNAL_BREAKPOINT,
368                     "Executed AML Breakpoint opcode");
369         break;
371     case AML_BREAK_OP:
372     case AML_CONTINUE_OP: /* ACPI 2.0 */
374         /* Pop and delete control states until we find a while */
376         while (WalkState->ControlState &&
377                (WalkState->ControlState->Control.Opcode != AML_WHILE_OP))
378         {
379             ControlState = AcpiUtPopGenericState (&WalkState->ControlState);
380             AcpiUtDeleteGenericState (ControlState);
```

```
381         }

383         /* No while found? */

385         if (!WalkState->ControlState)
386         {
387             return (AE_AML_NO_WHILE);
388         }

390         /* Was: WalkState->AmlLastWhile = WalkState->ControlState->Control.AmlPr

392         WalkState->AmlLastWhile = WalkState->ControlState->Control.PackageEnd;

394         /* Return status depending on opcode */

396         if (Op->Common.AmlOpcode == AML_BREAK_OP)
397         {
398             Status = AE_CTRL_BREAK;
399         }
400         else
401         {
402             Status = AE_CTRL_CONTINUE;
403         }
404         break;


406     default:

408         ACPI_ERROR ((AE_INFO, "Unknown control opcode=0x%X Op=%p",
409             Op->Common.AmlOpcode, Op));

411         Status = AE_AML_BAD_OPCODE;
412         break;
413     }

415     return (Status);
416 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   26047 Thu Dec 26 13:48:50 2013
new/usr/src/common/acpica/components/dispatcher/dsfield.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsfield - Dispatcher field routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSFIELD_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "amlcode.h"
  49 #include "acdispat.h"
  50 #include "acinterp.h"
  51 #include "acnamesp.h"
  52 #include "acparser.h"


  55 #define _COMPONENT          ACPI_DISPATCHER
  56         ACPI_MODULE_NAME    ("dsfield")

  58 /* Local prototypes */
```



```
  60 #ifdef ACPI_ASL_COMPILER
  61 #include "acdisasm.h"

  63 static ACPI_STATUS
  64 AcpiDsCreateExternalRegion (
  65     ACPI_STATUS             LookupStatus,
  66     ACPI_PARSE_OBJECT       *Op,
  67     char                    *Path,
  68     ACPI_WALK_STATE         *WalkState,
  69     ACPI_NAMESPACE_NODE     **Node);
  70 #endif

  72 static ACPI_STATUS
  73 AcpiDsGetFieldNames (
  74     ACPI_CREATE_FIELD_INFO  *Info,
  75     ACPI_WALK_STATE         *WalkState,
  76     ACPI_PARSE_OBJECT       *Arg);


  79 #ifdef ACPI_ASL_COMPILER
  80 /*******************************************************************************
  81  *
  82  * FUNCTION:    AcpiDsCreateExternalRegion (iASL Disassembler only)
  83  *
  84  * PARAMETERS:  LookupStatus    - Status from NsLookup operation
  85  *              Op              - Op containing the Field definition and args
  86  *              Path            - Pathname of the region
  87  *  ‘           WalkState       - Current method state
  88  *              Node            - Where the new region node is returned
  89  *
  90  * RETURN:      Status
  91  *
  92  * DESCRIPTION: Add region to the external list if NOT_FOUND. Create a new
  93  *              region node/object.
  94  *
  95  ******************************************************************************/

  97 static ACPI_STATUS
  98 AcpiDsCreateExternalRegion (
  99     ACPI_STATUS             LookupStatus,
 100     ACPI_PARSE_OBJECT       *Op,
 101     char                    *Path,
 102     ACPI_WALK_STATE         *WalkState,
 103     ACPI_NAMESPACE_NODE     **Node)
 104 {
 105     ACPI_STATUS             Status;
 106     ACPI_OPERAND_OBJECT     *ObjDesc;


 109     if (LookupStatus != AE_NOT_FOUND)
 110     {
 111         return (LookupStatus);
 112     }

 114     /*
 115      * Table disassembly:
 116      * OperationRegion not found. Generate an External for it, and
 117      * insert the name into the namespace.
 118      */
 119     AcpiDmAddOpToExternalList (Op, Path, ACPI_TYPE_REGION, 0, 0);
 120     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ACPI_TYPE_REGION,
 121         ACPI_IMODE_LOAD_PASS1, ACPI_NS_SEARCH_PARENT, WalkState, Node);
 122     if (ACPI_FAILURE (Status))
 123     {
 124         return (Status);
```

```
125     }

127     /* Must create and install a region object for the new node */

129     ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_REGION);
130     if (!ObjDesc)
131     {
132         return (AE_NO_MEMORY);
133     }

135     ObjDesc->Region.Node = *Node;
136     Status = AcpiNsAttachObject (*Node, ObjDesc, ACPI_TYPE_REGION);
137     return (Status);
138 }
139 #endif


142 /*****************************************************************************
143  *
144  * FUNCTION:    AcpiDsCreateBufferField
145  *
146  * PARAMETERS:  Op                  - Current parse op (CreateXXField)
147  *              WalkState           - Current state
148  *
149  * RETURN:      Status
150  *
151  * DESCRIPTION: Execute the CreateField operators:
152  *              CreateBitFieldOp,
153  *              CreateByteFieldOp,
154  *              CreateWordFieldOp,
155  *              CreateDwordFieldOp,
156  *              CreateQwordFieldOp,
 80  *              CreateDWordFieldOp,
 81  *              CreateQWordFieldOp,
157  *              CreateFieldOp       (all of which define a field in a buffer)
158  *
159  ****************************************************************************/

161 ACPI_STATUS
162 AcpiDsCreateBufferField (
163     ACPI_PARSE_OBJECT       *Op,
164     ACPI_WALK_STATE         *WalkState)
165 {
166     ACPI_PARSE_OBJECT       *Arg;
167     ACPI_NAMESPACE_NODE     *Node;
168     ACPI_STATUS             Status;
169     ACPI_OPERAND_OBJECT     *ObjDesc;
170     ACPI_OPERAND_OBJECT     *SecondDesc = NULL;
171     UINT32                  Flags;


174     ACPI_FUNCTION_TRACE (DsCreateBufferField);


177     /*
178      * Get the NameString argument (name of the new BufferField)
179      */
180     if (Op->Common.AmlOpcode == AML_CREATE_FIELD_OP)
181     {
182         /* For CreateField, name is the 4th argument */

184         Arg = AcpiPsGetArg (Op, 3);
185     }
186     else
187     {
188         /* For all other CreateXXXField operators, name is the 3rd argument */
```



```
190         Arg = AcpiPsGetArg (Op, 2);
191     }

193     if (!Arg)
194     {
195         return_ACPI_STATUS (AE_AML_NO_OPERAND);
196     }

198     if (WalkState->DeferredNode)
199     {
200         Node = WalkState->DeferredNode;
201         Status = AE_OK;
202     }
203     else
204     {
205         /* Execute flag should always be set when this function is entered */

207         if (!(WalkState->ParseFlags & ACPI_PARSE_EXECUTE))
208         {
209             return_ACPI_STATUS (AE_AML_INTERNAL);
210         }

212         /* Creating new namespace node, should not already exist */

214         Flags = ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE |
215                 ACPI_NS_ERROR_IF_FOUND;

217         /*
218          * Mark node temporary if we are executing a normal control
219          * method. (Don't mark if this is a module-level code method)
220          */
221         if (WalkState->MethodNode &&
222             !(WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
223         {
224             Flags |= ACPI_NS_TEMPORARY;
225         }

227         /* Enter the NameString into the namespace */

229         Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
230                     ACPI_TYPE_ANY, ACPI_IMODE_LOAD_PASS1,
231                     Flags, WalkState, &Node);
232         if (ACPI_FAILURE (Status))
233         {
234             ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
235             return_ACPI_STATUS (Status);
236         }
237     }

239     /*
240      * We could put the returned object (Node) on the object stack for later,
241      * but for now, we will put it in the "op" object that the parser uses,
242      * so we can get it again at the end of this scope.
243      */
244     Op->Common.Node = Node;

246     /*
247      * If there is no object attached to the node, this node was just created
248      * and we need to create the field object. Otherwise, this was a lookup
249      * of an existing node and we don't want to create the field object again.
250      */
251     ObjDesc = AcpiNsGetAttachedObject (Node);
252     if (ObjDesc)
253     {
254         return_ACPI_STATUS (AE_OK);
```

```
255     }
257     /*
258      * The Field definition is not fully parsed at this time.
259      * (We must save the address of the AML for the buffer and index operands)
260      */
262     /* Create the buffer field object */
264     ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_BUFFER_FIELD);
265     if (!ObjDesc)
266     {
267         Status = AE_NO_MEMORY;
268         goto Cleanup;
269     }
271     /*
272      * Remember location in AML stream of the field unit opcode and operands --
273      * since the buffer and index operands must be evaluated.
274      */
275     SecondDesc                = ObjDesc->Common.NextObject;
276     SecondDesc->Extra.AmlStart  = Op->Named.Data;
277     SecondDesc->Extra.AmlLength = Op->Named.Length;
278     ObjDesc->BufferField.Node   = Node;
280     /* Attach constructed field descriptors to parent node */
282     Status = AcpiNsAttachObject (Node, ObjDesc, ACPI_TYPE_BUFFER_FIELD);
283     if (ACPI_FAILURE (Status))
284     {
285         goto Cleanup;
286     }
289 Cleanup:
291     /* Remove local reference to the object */
293     AcpiUtRemoveReference (ObjDesc);
294     return_ACPI_STATUS (Status);
295 }
298 /*******************************************************************************
299  *
300  * FUNCTION:    AcpiDsGetFieldNames
301  *
302  * PARAMETERS:  Info            - CreateField info structure
303  *  '           WalkState       - Current method state
304  *              Arg             - First parser arg for the field name list
305  *
306  * RETURN:      Status
307  *
308  * DESCRIPTION: Process all named fields in a field declaration. Names are
309  *              entered into the namespace.
310  *
311  ******************************************************************************/
313 static ACPI_STATUS
314 AcpiDsGetFieldNames (
315     ACPI_CREATE_FIELD_INFO  *Info,
316     ACPI_WALK_STATE         *WalkState,
317     ACPI_PARSE_OBJECT       *Arg)
318 {
319     ACPI_STATUS             Status;
320     UINT64                  Position;
```



```
321     ACPI_PARSE_OBJECT       *Child;
324     ACPI_FUNCTION_TRACE_PTR (DsGetFieldNames, Info);
327     /* First field starts at bit zero */
329     Info->FieldBitPosition = 0;
331     /* Process all elements in the field list (of parse nodes) */
333     while (Arg)
334     {
335         /*
336          * Four types of field elements are handled:
337          * 1) Name - Enters a new named field into the namespace
338          * 2) Offset - specifies a bit offset
339          * 3) AccessAs - changes the access mode/attributes
340          * 4) Connection - Associate a resource template with the field
260          * Three types of field elements are handled:
261          * 1) Offset - specifies a bit offset
262          * 2) AccessAs - changes the access mode
263          * 3) Name - Enters a new named field into the namespace
341          */
342         switch (Arg->Common.AmlOpcode)
343         {
344         case AML_INT_RESERVEDFIELD_OP:
346             Position = (UINT64) Info->FieldBitPosition
347                         + (UINT64) Arg->Common.Value.Size;
349             if (Position > ACPI_UINT32_MAX)
350             {
351                 ACPI_ERROR ((AE_INFO,
352                     "Bit offset within field too large (> 0xFFFFFFFF)"));
353                 return_ACPI_STATUS (AE_SUPPORT);
354             }
356             Info->FieldBitPosition = (UINT32) Position;
357             break;
359         case AML_INT_ACCESSFIELD_OP:
360         case AML_INT_EXTACCESSFIELD_OP:
361             /*
362              * Get new AccessType, AccessAttribute, and AccessLength fields
363              * -- to be used for all field units that follow, until the
364              * end-of-field or another AccessAs keyword is encountered.
365              * NOTE. These three bytes are encoded in the integer value
366              * of the parseop for convenience.
286              * Get a new AccessType and AccessAttribute -- to be used for all
287              * field units that follow, until field end or another AccessAs
288              * keyword.
367              *
368              * In FieldFlags, preserve the flag bits other than the
369              * ACCESS_TYPE bits.
291              * ACCESS_TYPE bits
370              */
372             /* AccessType (ByteAcc, WordAcc, etc.) */
374             Info->FieldFlags = (UINT8)
375                 ((Info->FieldFlags & ~(AML_FIELD_ACCESS_TYPE_MASK)) |
376                 ((UINT8) ((UINT32) (Arg->Common.Value.Integer & 0x07))));
```

```
295                 ((UINT8) ((UINT32) Arg->Common.Value.Integer >> 8)));

378             /* AccessAttribute (AttribQuick, AttribByte, etc.) */

380             Info->Attribute = (UINT8) ((Arg->Common.Value.Integer >> 8) & 0xFF);

382             /* AccessLength (for serial/buffer protocols) */

384             Info->AccessLength = (UINT8) ((Arg->Common.Value.Integer >> 16) & 0x
297             Info->Attribute = (UINT8) (Arg->Common.Value.Integer);
385             break;

387         case AML_INT_CONNECTION_OP:
388             /*
389              * Clear any previous connection. New connection is used for all
390              * fields that follow, similar to AccessAs
391              */
392             Info->ResourceBuffer = NULL;
393             Info->ConnectionNode = NULL;

395             /*
396              * A Connection() is either an actual resource descriptor (buffer)
397              * or a named reference to a resource template
398              */
399             Child = Arg->Common.Value.Arg;
400             if (Child->Common.AmlOpcode == AML_INT_BYTELIST_OP)
401             {
402                 Info->ResourceBuffer = Child->Named.Data;
403                 Info->ResourceLength = (UINT16) Child->Named.Value.Integer;
404             }
405             else
406             {
407                 /* Lookup the Connection() namepath, it should already exist */

409                 Status = AcpiNsLookup (WalkState->ScopeInfo,
410                             Child->Common.Value.Name, ACPI_TYPE_ANY,
411                             ACPI_IMODE_EXECUTE, ACPI_NS_DONT_OPEN_SCOPE,
412                             WalkState, &Info->ConnectionNode);
413                 if (ACPI_FAILURE (Status))
414                 {
415                     ACPI_ERROR_NAMESPACE (Child->Common.Value.Name, Status);
416                     return_ACPI_STATUS (Status);
417                 }
418             }
419             break;

421         case AML_INT_NAMEDFIELD_OP:

423             /* Lookup the name, it should already exist */

425             Status = AcpiNsLookup (WalkState->ScopeInfo,
426                         (char *) &Arg->Named.Name, Info->FieldType,
427                         ACPI_IMODE_EXECUTE, ACPI_NS_DONT_OPEN_SCOPE,
428                         WalkState, &Info->FieldNode);
429             if (ACPI_FAILURE (Status))
430             {
431                 ACPI_ERROR_NAMESPACE ((char *) &Arg->Named.Name, Status);
432                 return_ACPI_STATUS (Status);
433             }
434             else
435             {
436                 Arg->Common.Node = Info->FieldNode;
437                 Info->FieldBitLength = Arg->Common.Value.Size;

439                 /*
440                  * If there is no object attached to the node, this node was
```



```
441                  * just created and we need to create the field object.
442                  * Otherwise, this was a lookup of an existing node and we
443                  * don't want to create the field object again.
444                  */
445                 if (!AcpiNsGetAttachedObject (Info->FieldNode))
446                 {
447                     Status = AcpiExPrepFieldValue (Info);
448                     if (ACPI_FAILURE (Status))
449                     {
450                         return_ACPI_STATUS (Status);
451                     }
452                 }
453             }

455             /* Keep track of bit position for the next field */

457             Position = (UINT64) Info->FieldBitPosition
458                         + (UINT64) Arg->Common.Value.Size;

460             if (Position > ACPI_UINT32_MAX)
461             {
462                 ACPI_ERROR ((AE_INFO,
463                     "Field [%4.4s] bit offset too large (> 0xFFFFFFFF)",
464                     ACPI_CAST_PTR (char, &Info->FieldNode->Name)));
465                 return_ACPI_STATUS (AE_SUPPORT);
466             }

468             Info->FieldBitPosition += Info->FieldBitLength;
469             break;

471         default:

473             ACPI_ERROR ((AE_INFO,
474                 "Invalid opcode in field list: 0x%X", Arg->Common.AmlOpcode));
475             return_ACPI_STATUS (AE_AML_BAD_OPCODE);
476         }

478         Arg = Arg->Common.Next;
479     }

481     return_ACPI_STATUS (AE_OK);
482 }


485 /*******************************************************************************
486  *
487  * FUNCTION:    AcpiDsCreateField
488  *
489  * PARAMETERS:  Op              - Op containing the Field definition and args
490  *              RegionNode      - Object for the containing Operation Region
491  *              WalkState       - Current method state
492  *
493  * RETURN:      Status
494  *
495  * DESCRIPTION: Create a new field in the specified operation region
496  *
497  ******************************************************************************/

499 ACPI_STATUS
500 AcpiDsCreateField (
501     ACPI_PARSE_OBJECT       *Op,
502     ACPI_NAMESPACE_NODE     *RegionNode,
503     ACPI_WALK_STATE         *WalkState)
504 {
505     ACPI_STATUS             Status;
```

```
    ACPI_PARSE_OBJECT       *Arg;
    ACPI_CREATE_FIELD_INFO  Info;


    ACPI_FUNCTION_TRACE_PTR (DsCreateField, Op);


    /* First arg is the name of the parent OpRegion (must already exist) */

    Arg = Op->Common.Value.Arg;

    if (!RegionNode)
    {
        Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.Name,
                    ACPI_TYPE_REGION, ACPI_IMODE_EXECUTE,
                    ACPI_NS_SEARCH_PARENT, WalkState, &RegionNode);
#ifdef ACPI_ASL_COMPILER
        Status = AcpiDsCreateExternalRegion (Status, Arg,
            Arg->Common.Value.Name, WalkState, &RegionNode);
#endif
        if (ACPI_FAILURE (Status))
        {
            ACPI_ERROR_NAMESPACE (Arg->Common.Value.Name, Status);
            return_ACPI_STATUS (Status);
        }
    }

    ACPI_MEMSET (&Info, 0, sizeof (ACPI_CREATE_FIELD_INFO));

    /* Second arg is the field flags */

    Arg = Arg->Common.Next;
    Info.FieldFlags = (UINT8) Arg->Common.Value.Integer;
    Info.Attribute = 0;

    /* Each remaining arg is a Named Field */

    Info.FieldType = ACPI_TYPE_LOCAL_REGION_FIELD;
    Info.RegionNode = RegionNode;

    Status = AcpiDsGetFieldNames (&Info, WalkState, Arg->Common.Next);

    return_ACPI_STATUS (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiDsInitFieldObjects
 *
 * PARAMETERS:  Op              - Op containing the Field definition and args
 *              WalkState       - Current method state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: For each "Field Unit" name in the argument list that is
 *              part of the field declaration, enter the name into the
 *              namespace.
 *
 ****************************************************************************/

ACPI_STATUS
AcpiDsInitFieldObjects (
    ACPI_PARSE_OBJECT       *Op,
    ACPI_WALK_STATE         *WalkState)
{
```



```
    ACPI_STATUS             Status;
    ACPI_PARSE_OBJECT       *Arg = NULL;
    ACPI_NAMESPACE_NODE     *Node;
    UINT8                   Type = 0;
    UINT32                  Flags;


    ACPI_FUNCTION_TRACE_PTR (DsInitFieldObjects, Op);


    /* Execute flag should always be set when this function is entered */

    if (!(WalkState->ParseFlags & ACPI_PARSE_EXECUTE))
    {
        if (WalkState->ParseFlags & ACPI_PARSE_DEFERRED_OP)
        {
            /* BankField Op is deferred, just return OK */

            return_ACPI_STATUS (AE_OK);
        }

        return_ACPI_STATUS (AE_AML_INTERNAL);
    }

    /*
     * Get the FieldList argument for this opcode. This is the start of the
     * list of field elements.
     */
    switch (WalkState->Opcode)
    {
    case AML_FIELD_OP:

        Arg = AcpiPsGetArg (Op, 2);
        Type = ACPI_TYPE_LOCAL_REGION_FIELD;
        break;

    case AML_BANK_FIELD_OP:

        Arg = AcpiPsGetArg (Op, 4);
        Type = ACPI_TYPE_LOCAL_BANK_FIELD;
        break;

    case AML_INDEX_FIELD_OP:

        Arg = AcpiPsGetArg (Op, 3);
        Type = ACPI_TYPE_LOCAL_INDEX_FIELD;
        break;

    default:

        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* Creating new namespace node(s), should not already exist */

    Flags = ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE |
            ACPI_NS_ERROR_IF_FOUND;

    /*
     * Mark node(s) temporary if we are executing a normal control
     * method. (Don't mark if this is a module-level code method)
     */
    if (WalkState->MethodNode &&
        !(WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
    {
        Flags |= ACPI_NS_TEMPORARY;
```

```
637     }

639     /*
640      * Walk the list of entries in the FieldList
641      * Note: FieldList can be of zero length. In this case, Arg will be NULL.
642      */
643     while (Arg)
644     {
645         /*
646          * Ignore OFFSET/ACCESSAS/CONNECTION terms here; we are only interested
647          * in the field names in order to enter them into the namespace.
517          * Ignore OFFSET and ACCESSAS terms here; we are only interested in the
518          * field names in order to enter them into the namespace.
648          */
649         if (Arg->Common.AmlOpcode == AML_INT_NAMEDFIELD_OP)
650         {
651             Status = AcpiNsLookup (WalkState->ScopeInfo,
652                         (char *) &Arg->Named.Name, Type, ACPI_IMODE_LOAD_PASS1,
653                         Flags, WalkState, &Node);
654             if (ACPI_FAILURE (Status))
655             {
656                 ACPI_ERROR_NAMESPACE ((char *) &Arg->Named.Name, Status);
657                 if (Status != AE_ALREADY_EXISTS)
658                 {
659                     return_ACPI_STATUS (Status);
660                 }

662                 /* Name already exists, just ignore this error */

664                 Status = AE_OK;
665             }

667             Arg->Common.Node = Node;
668         }

670         /* Get the next field element in the list */

672         Arg = Arg->Common.Next;
673     }

675     return_ACPI_STATUS (AE_OK);
676 }


679 /*******************************************************************************
680  *
681  * FUNCTION:    AcpiDsCreateBankField
682  *
683  * PARAMETERS:  Op              - Op containing the Field definition and args
684  *              RegionNode      - Object for the containing Operation Region
685  *              WalkState       - Current method state
686  *
687  * RETURN:      Status
688  *
689  * DESCRIPTION: Create a new bank field in the specified operation region
690  *
691  ******************************************************************************/

693 ACPI_STATUS
694 AcpiDsCreateBankField (
695     ACPI_PARSE_OBJECT       *Op,
696     ACPI_NAMESPACE_NODE     *RegionNode,
697     ACPI_WALK_STATE         *WalkState)
698 {
699     ACPI_STATUS             Status;
700     ACPI_PARSE_OBJECT       *Arg;
```



```
701     ACPI_CREATE_FIELD_INFO  Info;


704     ACPI_FUNCTION_TRACE_PTR (DsCreateBankField, Op);


707     /* First arg is the name of the parent OpRegion (must already exist) */

709     Arg = Op->Common.Value.Arg;
710     if (!RegionNode)
711     {
712         Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.Name,
713                     ACPI_TYPE_REGION, ACPI_IMODE_EXECUTE,
714                     ACPI_NS_SEARCH_PARENT, WalkState, &RegionNode);
715 #ifdef ACPI_ASL_COMPILER
716         Status = AcpiDsCreateExternalRegion (Status, Arg,
717             Arg->Common.Value.Name, WalkState, &RegionNode);
718 #endif
719         if (ACPI_FAILURE (Status))
720         {
721             ACPI_ERROR_NAMESPACE (Arg->Common.Value.Name, Status);
722             return_ACPI_STATUS (Status);
723         }
724     }

726     /* Second arg is the Bank Register (Field) (must already exist) */

728     Arg = Arg->Common.Next;
729     Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
730                     ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
731                     ACPI_NS_SEARCH_PARENT, WalkState, &Info.RegisterNode);
732     if (ACPI_FAILURE (Status))
733     {
734         ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
735         return_ACPI_STATUS (Status);
736     }

738     /*
739      * Third arg is the BankValue
740      * This arg is a TermArg, not a constant
741      * It will be evaluated later, by AcpiDsEvalBankFieldOperands
742      */
743     Arg = Arg->Common.Next;

745     /* Fourth arg is the field flags */

747     Arg = Arg->Common.Next;
748     Info.FieldFlags = (UINT8) Arg->Common.Value.Integer;

750     /* Each remaining arg is a Named Field */

752     Info.FieldType = ACPI_TYPE_LOCAL_BANK_FIELD;
753     Info.RegionNode = RegionNode;

755     /*
756      * Use Info.DataRegisterNode to store BankField Op
757      * It’s safe because DataRegisterNode will never be used when create bank fi
758      * We store AmlStart and AmlLength in the BankField Op for late evaluation
759      * Used in AcpiExPrepFieldValue(Info)
760      *
761      * TBD: Or, should we add a field in ACPI_CREATE_FIELD_INFO, like "void *Par
762      */
763     Info.DataRegisterNode = (ACPI_NAMESPACE_NODE*) Op;

765     Status = AcpiDsGetFieldNames (&Info, WalkState, Arg->Common.Next);
766     return_ACPI_STATUS (Status);
```

```
 767 }


 770 /*****************************************************************************
 771  *
 772  * FUNCTION:    AcpiDsCreateIndexField
 773  *
 774  * PARAMETERS:  Op              - Op containing the Field definition and args
 775  *              RegionNode      - Object for the containing Operation Region
 776  *  ‘           WalkState       - Current method state
 777  *
 778  * RETURN:      Status
 779  *
 780  * DESCRIPTION: Create a new index field in the specified operation region
 781  *
 782  ****************************************************************************/

 784 ACPI_STATUS
 785 AcpiDsCreateIndexField (
 786     ACPI_PARSE_OBJECT       *Op,
 787     ACPI_NAMESPACE_NODE     *RegionNode,
 788     ACPI_WALK_STATE         *WalkState)
 789 {
 790     ACPI_STATUS             Status;
 791     ACPI_PARSE_OBJECT       *Arg;
 792     ACPI_CREATE_FIELD_INFO  Info;


 795     ACPI_FUNCTION_TRACE_PTR (DsCreateIndexField, Op);


 798     /* First arg is the name of the Index register (must already exist) */

 800     Arg = Op->Common.Value.Arg;
 801     Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
 802                     ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
 803                     ACPI_NS_SEARCH_PARENT, WalkState, &Info.RegisterNode);
 804     if (ACPI_FAILURE (Status))
 805     {
 806         ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
 807         return_ACPI_STATUS (Status);
 808     }

 810     /* Second arg is the data register (must already exist) */

 812     Arg = Arg->Common.Next;
 813     Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
 814                     ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
 815                     ACPI_NS_SEARCH_PARENT, WalkState, &Info.DataRegisterNode);
 816     if (ACPI_FAILURE (Status))
 817     {
 818         ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
 819         return_ACPI_STATUS (Status);
 820     }

 822     /* Next arg is the field flags */

 824     Arg = Arg->Common.Next;
 825     Info.FieldFlags = (UINT8) Arg->Common.Value.Integer;

 827     /* Each remaining arg is a Named Field */

 829     Info.FieldType = ACPI_TYPE_LOCAL_INDEX_FIELD;
 830     Info.RegionNode = RegionNode;

 832     Status = AcpiDsGetFieldNames (&Info, WalkState, Arg->Common.Next);
```



```
 833     return_ACPI_STATUS (Status);
 834 }
```

```
**********************************************************
    7212 Thu Dec 26 13:48:50 2013
new/usr/src/common/acpica/components/dispatcher/dsinit.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsinit - Object initialization namespace walk
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSINIT_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acdispat.h"
  49 #include "acnamesp.h"
  50 #include "actables.h"

  52 #define _COMPONENT          ACPI_DISPATCHER
  53         ACPI_MODULE_NAME    ("dsinit")

  55 /* Local prototypes */

  57 static ACPI_STATUS
  58 AcpiDsInitOneObject (
  59     ACPI_HANDLE             ObjHandle,
```



```
  60     UINT32                  Level,
  61     void                    *Context,
  62     void                    **ReturnValue);


  65 /******************************************************************************
  66  *
  67  * FUNCTION:    AcpiDsInitOneObject
  68  *
  69  * PARAMETERS:  ObjHandle       - Node for the object
  70  *              Level           - Current nesting level
  71  *              Context         - Points to a init info struct
  72  *              ReturnValue     - Not used
  73  *
  74  * RETURN:      Status
  75  *
  76  * DESCRIPTION: Callback from AcpiWalkNamespace. Invoked for every object
  77  *              within the namespace.
  78  *
  79  *              Currently, the only objects that require initialization are:
  80  *              1) Methods
  81  *              2) Operation Regions
  82  *
  83  *****************************************************************************/

  85 static ACPI_STATUS
  86 AcpiDsInitOneObject (
  87     ACPI_HANDLE             ObjHandle,
  88     UINT32                  Level,
  89     void                    *Context,
  90     void                    **ReturnValue)
  91 {
  92     ACPI_INIT_WALK_INFO     *Info = (ACPI_INIT_WALK_INFO *) Context;
  93     ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
  94     ACPI_OBJECT_TYPE        Type;
  95     ACPI_STATUS             Status;


  98     ACPI_FUNCTION_ENTRY ();


 101     /*
 102      * We are only interested in NS nodes owned by the table that
 103      * was just loaded
 104      */
 105     if (Node->OwnerId != Info->OwnerId)
 106     {
 107         return (AE_OK);
 108     }

 110     Info->ObjectCount++;

 112     /* And even then, we are only interested in a few object types */

 114     Type = AcpiNsGetType (ObjHandle);

 116     switch (Type)
 117     {
 118     case ACPI_TYPE_REGION:

 120         Status = AcpiDsInitializeRegion (ObjHandle);
 121         if (ACPI_FAILURE (Status))
 122         {
 123             ACPI_EXCEPTION ((AE_INFO, Status,
 124                 "During Region initialization %p [%4.4s]",
 125                 ObjHandle, AcpiUtGetNodeName (ObjHandle)));
```

```
126         }
128         Info->OpRegionCount++;
129         break;

131     case ACPI_TYPE_METHOD:
133         Info->MethodCount++;
134         break;

136     case ACPI_TYPE_DEVICE:
138         Info->DeviceCount++;
139         break;

141     default:

144     default:
143         break;
144     }

146     /*
147      * We ignore errors from above, and always return OK, since
148      * we don't want to abort the walk on a single error.
149      */
150     return (AE_OK);
151 }
```

unchanged_portion_omitted_

```
**********************************************************
   23030 Thu Dec 26 13:48:51 2013
new/usr/src/common/acpica/components/dispatcher/dsmethod.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsmethod - Parser/Interpreter interface - control method parsing
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSMETHOD_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acdispat.h"
  49 #include "acinterp.h"
  50 #include "acnamesp.h"
  51 #include "acdisasm.h"


  54 #define _COMPONENT          ACPI_DISPATCHER
  55         ACPI_MODULE_NAME    ("dsmethod")

  57 /* Local prototypes */

  59 static ACPI_STATUS
```



```
  60 AcpiDsCreateMethodMutex (
  61     ACPI_OPERAND_OBJECT     *MethodDesc);


  64 /******************************************************************************
  65  *
  66  * FUNCTION:    AcpiDsMethodError
  67  *
  68  * PARAMETERS:  Status          - Execution status
  69  *              WalkState       - Current state
  70  *
  71  * RETURN:      Status
  72  *
  73  * DESCRIPTION: Called on method error. Invoke the global exception handler if
  74  *              present, dump the method data if the disassembler is configured
  75  *
  76  *              Note: Allows the exception handler to change the status code
  77  *
  78  *****************************************************************************/

  80 ACPI_STATUS
  81 AcpiDsMethodError (
  82     ACPI_STATUS             Status,
  83     ACPI_WALK_STATE         *WalkState)
  84 {
  85     ACPI_FUNCTION_ENTRY ();


  88     /* Ignore AE_OK and control exception codes */

  90     if (ACPI_SUCCESS (Status) ||
  91         (Status & AE_CODE_CONTROL))
  92     {
  93         return (Status);
  94     }

  96     /* Invoke the global exception handler */

  98     if (AcpiGbl_ExceptionHandler)
  99     {
 100         /* Exit the interpreter, allow handler to execute methods */

 102         AcpiExExitInterpreter ();

 104         /*
 105          * Handler can map the exception code to anything it wants, including
 106          * AE_OK, in which case the executing method will not be aborted.
 107          */
 108         Status = AcpiGbl_ExceptionHandler (Status,
 109                     WalkState->MethodNode ?
 110                         WalkState->MethodNode->Name.Integer : 0,
 111                     WalkState->Opcode, WalkState->AmlOffset, NULL);
 112         AcpiExEnterInterpreter ();
 113     }

 115     AcpiDsClearImplicitReturn (WalkState);

 117 #ifdef ACPI_DISASSEMBLER
 118     if (ACPI_FAILURE (Status))
 119     {
 120         /* Display method locals/args if disassembler is present */

 122         AcpiDmDumpMethodInfo (Status, WalkState, WalkState->Op);
 123     }
 124 #endif
```

```
126     return (Status);
127 }


130 /*****************************************************************************
131  *
132  * FUNCTION:    AcpiDsCreateMethodMutex
133  *
134  * PARAMETERS:  ObjDesc             - The method object
135  *
136  * RETURN:      Status
137  *
138  * DESCRIPTION: Create a mutex object for a serialized control method
139  *
140  ****************************************************************************/

142 static ACPI_STATUS
143 AcpiDsCreateMethodMutex (
144     ACPI_OPERAND_OBJECT     *MethodDesc)
145 {
146     ACPI_OPERAND_OBJECT     *MutexDesc;
147     ACPI_STATUS             Status;


150     ACPI_FUNCTION_TRACE (DsCreateMethodMutex);


153     /* Create the new mutex object */

155     MutexDesc = AcpiUtCreateInternalObject (ACPI_TYPE_MUTEX);
156     if (!MutexDesc)
157     {
158         return_ACPI_STATUS (AE_NO_MEMORY);
159     }

161     /* Create the actual OS Mutex */

163     Status = AcpiOsCreateMutex (&MutexDesc->Mutex.OsMutex);
164     if (ACPI_FAILURE (Status))
165     {
166         AcpiUtDeleteObjectDesc (MutexDesc);
167         return_ACPI_STATUS (Status);
168     }

170     MutexDesc->Mutex.SyncLevel = MethodDesc->Method.SyncLevel;
171     MethodDesc->Method.Mutex = MutexDesc;
172     return_ACPI_STATUS (AE_OK);
173 }
```

*unchanged_portion_omitted_*

```
331 /*****************************************************************************
332  *
333  * FUNCTION:    AcpiDsCallControlMethod
334  *
335  * PARAMETERS:  Thread              - Info for this thread
336  *              ThisWalkState       - Current walk state
337  *              Op                  - Current Op to be walked
338  *
339  * RETURN:      Status
340  *
341  * DESCRIPTION: Transfer execution to a called control method
342  *
343  ****************************************************************************/

345 ACPI_STATUS
```



```
346 AcpiDsCallControlMethod (
347     ACPI_THREAD_STATE       *Thread,
348     ACPI_WALK_STATE         *ThisWalkState,
349     ACPI_PARSE_OBJECT       *Op)
350 {
351     ACPI_STATUS             Status;
352     ACPI_NAMESPACE_NODE     *MethodNode;
353     ACPI_WALK_STATE         *NextWalkState = NULL;
354     ACPI_OPERAND_OBJECT     *ObjDesc;
355     ACPI_EVALUATE_INFO      *Info;
356     UINT32                  i;


359     ACPI_FUNCTION_TRACE_PTR (DsCallControlMethod, ThisWalkState);

361     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Calling method %p, currentstate=%p\n",
362         ThisWalkState->PrevOp, ThisWalkState));

364     /*
365      * Get the namespace entry for the control method we are about to call
366      */
367     MethodNode = ThisWalkState->MethodCallNode;
368     if (!MethodNode)
369     {
370         return_ACPI_STATUS (AE_NULL_ENTRY);
371     }

373     ObjDesc = AcpiNsGetAttachedObject (MethodNode);
374     if (!ObjDesc)
375     {
376         return_ACPI_STATUS (AE_NULL_OBJECT);
377     }

379     /* Init for new method, possibly wait on method mutex */

381     Status = AcpiDsBeginMethodExecution (MethodNode, ObjDesc,
382                 ThisWalkState);
383     if (ACPI_FAILURE (Status))
384     {
385         return_ACPI_STATUS (Status);
386     }

388     /* Begin method parse/execution. Create a new walk state */

390     NextWalkState = AcpiDsCreateWalkState (ObjDesc->Method.OwnerId,
391                         NULL, ObjDesc, Thread);
392     if (!NextWalkState)
393     {
394         Status = AE_NO_MEMORY;
395         goto Cleanup;
396     }

398     /*
399      * The resolved arguments were put on the previous walk state's operand
400      * stack. Operands on the previous walk state stack always
401      * start at index 0. Also, null terminate the list of arguments
402      */
403     ThisWalkState->Operands [ThisWalkState->NumOperands] = NULL;

405     /*
406      * Allocate and initialize the evaluation information block
407      * TBD: this is somewhat inefficient, should change interface to
408      * DsInitAmlWalk. For now, keeps this struct off the CPU stack
409      */
410     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
411     if (!Info)
```

```
412     {
413         Status = AE_NO_MEMORY;
414         goto Cleanup;
412         return_ACPI_STATUS (AE_NO_MEMORY);
415     }

417     Info->Parameters = &ThisWalkState->Operands[0];

419     Status = AcpiDsInitAmlWalk (NextWalkState, NULL, MethodNode,
420                 ObjDesc->Method.AmlStart, ObjDesc->Method.AmlLength,
421                 Info, ACPI_IMODE_EXECUTE);

423     ACPI_FREE (Info);
424     if (ACPI_FAILURE (Status))
425     {
426         goto Cleanup;
427     }

429     /*
430      * Delete the operands on the previous walkstate operand stack
431      * (they were copied to new objects)
432      */
433     for (i = 0; i < ObjDesc->Method.ParamCount; i++)
434     {
435         AcpiUtRemoveReference (ThisWalkState->Operands [i]);
436         ThisWalkState->Operands [i] = NULL;
437     }

439     /* Clear the operand stack */

441     ThisWalkState->NumOperands = 0;

443     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
444         "**** Begin nested execution of [%4.4s] **** WalkState=%p\n",
445         MethodNode->Name.Ascii, NextWalkState));

447     /* Invoke an internal method if necessary */

449     if (ObjDesc->Method.InfoFlags & ACPI_METHOD_INTERNAL_ONLY)
450     {
451         Status = ObjDesc->Method.Dispatch.Implementation (NextWalkState);
452         if (Status == AE_OK)
453         {
454             Status = AE_CTRL_TERMINATE;
455         }
456     }

458     return_ACPI_STATUS (Status);


461 Cleanup:

463     /* On error, we must terminate the method properly */

465     AcpiDsTerminateControlMethod (ObjDesc, NextWalkState);
466     if (NextWalkState)
467     {
468         AcpiDsDeleteWalkState (NextWalkState);
469     }

471     return_ACPI_STATUS (Status);
472 }
```

_____***unchanged_portion_omitted_***

****
23466 Thu Dec 26 13:48:51 2013
new/usr/src/common/acpica/components/dispatcher/dsmthdat.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dsmthdat - control method arguments and local variables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DSMTHDAT_C__

#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"
#include "acnamesp.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_DISPATCHER
        ACPI_MODULE_NAME    ("dsmthdat")

/* Local prototypes */

static void
AcpiDsMethodDataDeleteValue (
```



```
    UINT8                   Type,
    UINT32                  Index,
    ACPI_WALK_STATE         *WalkState);

static ACPI_STATUS
AcpiDsMethodDataSetValue (
    UINT8                   Type,
    UINT32                  Index,
    ACPI_OPERAND_OBJECT     *Object,
    ACPI_WALK_STATE         *WalkState);

#ifdef ACPI_OBSOLETE_FUNCTIONS
ACPI_OBJECT_TYPE
AcpiDsMethodDataGetType (
    UINT16                  Opcode,
    UINT32                  Index,
    ACPI_WALK_STATE         *WalkState);
#endif


/******************************************************************************
 *
 * FUNCTION:    AcpiDsMethodDataInit
 *
 * PARAMETERS:  WalkState           - Current walk state object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize the data structures that hold the method's arguments
 *              and locals. The data struct is an array of namespace nodes for
 *              each - this allows RefOf and DeRefOf to work properly for these
 *              special data types.
 *
 * NOTES:       WalkState fields are initialized to zero by the
 *              ACPI_ALLOCATE_ZEROED().
 *
 *              A pseudo-Namespace Node is assigned to each argument and local
 *              so that RefOf() can return a pointer to the Node.
 *
 *****************************************************************************/

void
AcpiDsMethodDataInit (
    ACPI_WALK_STATE         *WalkState)
{
    UINT32                  i;


    ACPI_FUNCTION_TRACE (DsMethodDataInit);


    /* Init the method arguments */

    for (i = 0; i < ACPI_METHOD_NUM_ARGS; i++)
    {
        ACPI_MOVE_32_TO_32 (&WalkState->Arguments[i].Name, NAMEOF_ARG_NTE);
        WalkState->Arguments[i].Name.Integer |= (i << 24);
        WalkState->Arguments[i].DescriptorType = ACPI_DESC_TYPE_NAMED;
        WalkState->Arguments[i].Type = ACPI_TYPE_ANY;
        WalkState->Arguments[i].Flags = ANOBJ_METHOD_ARG;
    }

    /* Init the method locals */

    for (i = 0; i < ACPI_METHOD_NUM_LOCALS; i++)
    {
```

```
126         ACPI_MOVE_32_TO_32 (&WalkState->LocalVariables[i].Name, NAMEOF_LOCAL_NTE

128         WalkState->LocalVariables[i].Name.Integer |= (i << 24);
129         WalkState->LocalVariables[i].DescriptorType = ACPI_DESC_TYPE_NAMED;
130         WalkState->LocalVariables[i].Type = ACPI_TYPE_ANY;
131         WalkState->LocalVariables[i].Flags = ANOBJ_METHOD_LOCAL;
132     }

134     return_VOID;
135 }
```

*unchanged_portion_omitted_*

```
256 /*****************************************************************************
257  *
258  * FUNCTION:    AcpiDsMethodDataGetNode
259  *
260  * PARAMETERS:  Type                - Either ACPI_REFCLASS_LOCAL or
261  *                                    ACPI_REFCLASS_ARG
262  *              Index               - Which Local or Arg whose type to get
263  *              WalkState           - Current walk state object
264  *              Node                - Where the node is returned.
265  *
266  * RETURN:      Status and node
267  *
268  * DESCRIPTION: Get the Node associated with a local or arg.
269  *
270  ****************************************************************************/

272 ACPI_STATUS
273 AcpiDsMethodDataGetNode (
274     UINT8                   Type,
275     UINT32                  Index,
276     ACPI_WALK_STATE         *WalkState,
277     ACPI_NAMESPACE_NODE     **Node)
278 {
279     ACPI_FUNCTION_TRACE (DsMethodDataGetNode);


282     /*
283      * Method Locals and Arguments are supported
284      */
285     switch (Type)
286     {
287     case ACPI_REFCLASS_LOCAL:

289         if (Index > ACPI_METHOD_MAX_LOCAL)
290         {
291             ACPI_ERROR ((AE_INFO,
292                 "Local index %u is invalid (max %u)",
293                 Index, ACPI_METHOD_MAX_LOCAL));
294             return_ACPI_STATUS (AE_AML_INVALID_INDEX);
295         }

297         /* Return a pointer to the pseudo-node */

299         *Node = &WalkState->LocalVariables[Index];
300         break;

302     case ACPI_REFCLASS_ARG:

304         if (Index > ACPI_METHOD_MAX_ARG)
305         {
306             ACPI_ERROR ((AE_INFO,
307                 "Arg index %u is invalid (max %u)",
308                 Index, ACPI_METHOD_MAX_ARG));
```



```
309             return_ACPI_STATUS (AE_AML_INVALID_INDEX);
310         }

312         /* Return a pointer to the pseudo-node */

314         *Node = &WalkState->Arguments[Index];
315         break;

317     default:

319         ACPI_ERROR ((AE_INFO, "Type %u is invalid", Type));
320         return_ACPI_STATUS (AE_TYPE);
321     }

323     return_ACPI_STATUS (AE_OK);
324 }
```

*unchanged_portion_omitted_*

```
386 /*****************************************************************************
387  *
388  * FUNCTION:    AcpiDsMethodDataGetValue
389  *
390  * PARAMETERS:  Type                - Either ACPI_REFCLASS_LOCAL or
391  *                                    ACPI_REFCLASS_ARG
392  *              Index               - Which localVar or argument to get
393  *              WalkState           - Current walk state object
394  *              DestDesc            - Where Arg or Local value is returned
395  *
396  * RETURN:      Status
397  *
398  * DESCRIPTION: Retrieve value of selected Arg or Local for this method
399  *              Used only in AcpiExResolveToValue().
400  *
401  ****************************************************************************/

403 ACPI_STATUS
404 AcpiDsMethodDataGetValue (
405     UINT8                   Type,
406     UINT32                  Index,
407     ACPI_WALK_STATE         *WalkState,
408     ACPI_OPERAND_OBJECT     **DestDesc)
409 {
410     ACPI_STATUS             Status;
411     ACPI_NAMESPACE_NODE     *Node;
412     ACPI_OPERAND_OBJECT     *Object;


415     ACPI_FUNCTION_TRACE (DsMethodDataGetValue);


418     /* Validate the object descriptor */

420     if (!DestDesc)
421     {
422         ACPI_ERROR ((AE_INFO, "Null object descriptor pointer"));
423         return_ACPI_STATUS (AE_BAD_PARAMETER);
424     }

426     /* Get the namespace node for the arg/local */

428     Status = AcpiDsMethodDataGetNode (Type, Index, WalkState, &Node);
429     if (ACPI_FAILURE (Status))
430     {
431         return_ACPI_STATUS (Status);
432     }
```

```
434     /* Get the object from the node */

436     Object = Node->Object;

438     /* Examine the returned object, it must be valid. */

440     if (!Object)
441     {
442         /*
443          * Index points to uninitialized object.
444          * This means that either 1) The expected argument was
445          * not passed to the method, or 2) A local variable
446          * was referenced by the method (via the ASL)
447          * before it was initialized. Either case is an error.
448          */

450         /* If slack enabled, init the LocalX/ArgX to an Integer of value zero */

452         if (AcpiGbl_EnableInterpreterSlack)
453         {
454             Object = AcpiUtCreateIntegerObject ((UINT64) 0);
455             if (!Object)
456             {
457                 return_ACPI_STATUS (AE_NO_MEMORY);
458             }

460             Node->Object = Object;
461         }

463         /* Otherwise, return the error */

465         else switch (Type)
466         {
467         case ACPI_REFCLASS_ARG:

469             ACPI_ERROR ((AE_INFO,
470                 "Uninitialized Arg[%u] at node %p",
471                 Index, Node));

473             return_ACPI_STATUS (AE_AML_UNINITIALIZED_ARG);

475         case ACPI_REFCLASS_LOCAL:

476             /*
477              * No error message for this case, will be trapped again later to
478              * detect and ignore cases of Store(LocalX,LocalX)
479              */
480             return_ACPI_STATUS (AE_AML_UNINITIALIZED_LOCAL);

482         default:

484             ACPI_ERROR ((AE_INFO, "Not a Arg/Local opcode: 0x%X", Type));
485             return_ACPI_STATUS (AE_AML_INTERNAL);
486         }
487     }

489     /*
490      * The Index points to an initialized and valid object.
491      * Return an additional reference to the object
492      */
493     *DestDesc = Object;
494     AcpiUtAddReference (Object);

496     return_ACPI_STATUS (AE_OK);
497 }
```

*unchanged_portion_omitted_*


```
770 #endif
```

```
**********************************************************
   28342 Thu Dec 26 13:48:52 2013
new/usr/src/common/acpica/components/dispatcher/dsobject.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsobject - Dispatcher object management routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSOBJECT_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acparser.h"
  49 #include "amlcode.h"
  50 #include "acdispat.h"
  51 #include "acnamesp.h"
  52 #include "acinterp.h"

  54 #define _COMPONENT          ACPI_DISPATCHER
  55         ACPI_MODULE_NAME    ("dsobject")

  57 /* Local prototypes */

  59 static ACPI_STATUS
```



```
  60 AcpiDsBuildInternalObject (
  61     ACPI_WALK_STATE         *WalkState,
  62     ACPI_PARSE_OBJECT       *Op,
  63     ACPI_OPERAND_OBJECT     **ObjDescPtr);


  66 #ifndef ACPI_NO_METHOD_EXECUTION
  67 /*****************************************************************************
  68  *
  69  * FUNCTION:    AcpiDsBuildInternalObject
  70  *
  71  * PARAMETERS:  WalkState       - Current walk state
  72  *              Op              - Parser object to be translated
  73  *              ObjDescPtr      - Where the ACPI internal object is returned
  74  *
  75  * RETURN:      Status
  76  *
  77  * DESCRIPTION: Translate a parser Op object to the equivalent namespace object
  78  *              Simple objects are any objects other than a package object!
  79  *
  80  ****************************************************************************/

  82 static ACPI_STATUS
  83 AcpiDsBuildInternalObject (
  84     ACPI_WALK_STATE         *WalkState,
  85     ACPI_PARSE_OBJECT       *Op,
  86     ACPI_OPERAND_OBJECT     **ObjDescPtr)
  87 {
  88     ACPI_OPERAND_OBJECT     *ObjDesc;
  89     ACPI_STATUS             Status;
  90     ACPI_OBJECT_TYPE        Type;


  93     ACPI_FUNCTION_TRACE (DsBuildInternalObject);


  96     *ObjDescPtr = NULL;
  97     if (Op->Common.AmlOpcode == AML_INT_NAMEPATH_OP)
  98     {
  99         /*
 100          * This is a named object reference. If this name was
 101          * previously looked up in the namespace, it was stored in this op.
 102          * Otherwise, go ahead and look it up now
 103          */
 104         if (!Op->Common.Node)
 105         {
 106             Status = AcpiNsLookup (WalkState->ScopeInfo,
 107                         Op->Common.Value.String,
 108                         ACPI_TYPE_ANY, ACPI_IMODE_EXECUTE,
 109                         ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPEN_SCOPE, NULL,
 110                         ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE, &(Op->Commo
 111             if (ACPI_FAILURE (Status))
 112             {
 113                 /* Check if we are resolving a named reference within a package

 115                 if ((Status == AE_NOT_FOUND) && (AcpiGbl_EnableInterpreterSlack)

 117                     ((Op->Common.Parent->Common.AmlOpcode == AML_PACKAGE_OP) ||
 118                      (Op->Common.Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP)
 119                 {
 120                     /*
 121                      * We didn't find the target and we are populating elements
 122                      * of a package - ignore if slack enabled. Some ASL code
 123                      * contains dangling invalid references in packages and
 124                      * expects that no exception will be issued. Leave the
 125                      * element as a null element. It cannot be used, but it
```

```
                     * can be overwritten by subsequent ASL code - this is
                     * typically the case.
                     */
                    ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
                        "Ignoring unresolved reference in package [%4.4s]\n",
                        WalkState->ScopeInfo->Scope.Node->Name.Ascii));

                    return_ACPI_STATUS (AE_OK);
                }
                else
                {
                    ACPI_ERROR_NAMESPACE (Op->Common.Value.String, Status);
                }

                return_ACPI_STATUS (Status);
            }
        }

        /* Special object resolution for elements of a package */

        if ((Op->Common.Parent->Common.AmlOpcode == AML_PACKAGE_OP) ||
            (Op->Common.Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
        {
            /*
             * Attempt to resolve the node to a value before we insert it into
             * the package. If this is a reference to a common data type,
             * resolve it immediately. According to the ACPI spec, package
             * elements can only be "data objects" or method references.
             * Attempt to resolve to an Integer, Buffer, String or Package.
             * If cannot, return the named reference (for things like Devices,
             * Methods, etc.) Buffer Fields and Fields will resolve to simple
             * objects (int/buf/str/pkg).
             *
             * NOTE: References to things like Devices, Methods, Mutexes, etc.
             * will remain as named references. This behavior is not described
             * in the ACPI spec, but it appears to be an oversight.
             */
            ObjDesc = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Op->Common.Node);

            Status = AcpiExResolveNodeToValue (
                        ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE, &ObjDesc),
                        WalkState);
            if (ACPI_FAILURE (Status))
            {
                return_ACPI_STATUS (Status);
            }

            /*
             * Special handling for Alias objects. We need to setup the type
             * and the Op->Common.Node to point to the Alias target. Note,
             * Alias has at most one level of indirection internally.
             */
            Type = Op->Common.Node->Type;
            if (Type == ACPI_TYPE_LOCAL_ALIAS)
            {
                Type = ObjDesc->Common.Type;
                Op->Common.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
                    Op->Common.Node->Object);
            }

            switch (Type)
            {
            /*
             * For these types, we need the actual node, not the subobject.
             * However, the subobject did not get an extra reference count above
             *
```



```
             * TBD: should ExResolveNodeToValue be changed to fix this?
             */
            case ACPI_TYPE_DEVICE:
            case ACPI_TYPE_THERMAL:

                AcpiUtAddReference (Op->Common.Node->Object);

                /*lint -fallthrough */
            /*
             * For these types, we need the actual node, not the subobject.
             * The subobject got an extra reference count in ExResolveNodeToValu
             */
            case ACPI_TYPE_MUTEX:
            case ACPI_TYPE_METHOD:
            case ACPI_TYPE_POWER:
            case ACPI_TYPE_PROCESSOR:
            case ACPI_TYPE_EVENT:
            case ACPI_TYPE_REGION:

                /* We will create a reference object for these types below */
                break;

            default:
                /*
                 * All other types - the node was resolved to an actual
                 * object, we are done.
                 */
                goto Exit;
            }
        }
    }

    /* Create and init a new internal ACPI object */

    ObjDesc = AcpiUtCreateInternalObject (
                (AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode))->ObjectType);
    if (!ObjDesc)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    Status = AcpiDsInitObjectFromOp (WalkState, Op, Op->Common.AmlOpcode,
                &ObjDesc);
    if (ACPI_FAILURE (Status))
    {
        AcpiUtRemoveReference (ObjDesc);
        return_ACPI_STATUS (Status);
    }

Exit:
    *ObjDescPtr = ObjDesc;
    return_ACPI_STATUS (Status);
}
```

unchanged_portion_omitted_

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiDsBuildInternalPackageObj
 *
 * PARAMETERS:  WalkState       - Current walk state
 *              Op              - Parser object to be translated
 *              ElementCount    - Number of elements in the package - this is
 *                                the NumElements argument to Package()
 *              ObjDescPtr      - Where the ACPI internal object is returned
 *
```

```
 374  * RETURN:      Status
 375  *
 376  * DESCRIPTION: Translate a parser Op package object to the equivalent
 377  *              namespace object
 378  *
 379  * NOTE: The number of elements in the package will be always be the NumElements
 380  * count, regardless of the number of elements in the package list. If
 381  * NumElements is smaller, only that many package list elements are used.
 382  * if NumElements is larger, the Package object is padded out with
 383  * objects of type Uninitialized (as per ACPI spec.)
 384  *
 385  * Even though the ASL compilers do not allow NumElements to be smaller
 386  * than the Package list length (for the fixed length package opcode), some
 387  * BIOS code modifies the AML on the fly to adjust the NumElements, and
 388  * this code compensates for that. This also provides compatibility with
 389  * other AML interpreters.
 390  *
 391  *****************************************************************************/

 393 ACPI_STATUS
 394 AcpiDsBuildInternalPackageObj (
 395     ACPI_WALK_STATE         *WalkState,
 396     ACPI_PARSE_OBJECT       *Op,
 397     UINT32                  ElementCount,
 398     ACPI_OPERAND_OBJECT     **ObjDescPtr)
 399 {
 400     ACPI_PARSE_OBJECT       *Arg;
 401     ACPI_PARSE_OBJECT       *Parent;
 402     ACPI_OPERAND_OBJECT     *ObjDesc = NULL;
 403     ACPI_STATUS             Status = AE_OK;
 404     UINT32                  i;
 405     UINT16                  Index;
 406     UINT16                  ReferenceCount;


 409     ACPI_FUNCTION_TRACE (DsBuildInternalPackageObj);


 412     /* Find the parent of a possibly nested package */

 414     Parent = Op->Common.Parent;
 415     while ((Parent->Common.AmlOpcode == AML_PACKAGE_OP) ||
 416            (Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
 417     {
 418         Parent = Parent->Common.Parent;
 419     }

 421     /*
 422      * If we are evaluating a Named package object "Name (xxxx, Package)",
 423      * the package object already exists, otherwise it must be created.
 424      */
 425     ObjDesc = *ObjDescPtr;
 426     if (!ObjDesc)
 427     {
 428         ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_PACKAGE);
 429         *ObjDescPtr = ObjDesc;
 430         if (!ObjDesc)
 431         {
 432             return_ACPI_STATUS (AE_NO_MEMORY);
 433         }

 435         ObjDesc->Package.Node = Parent->Common.Node;
 436     }

 438     /*
 439      * Allocate the element array (array of pointers to the individual
```



```
 440      * objects) based on the NumElements parameter. Add an extra pointer slot
 441      * so that the list is always null terminated.
 442      */
 443     ObjDesc->Package.Elements = ACPI_ALLOCATE_ZEROED (
 444         ((ACPI_SIZE) ElementCount + 1) * sizeof (void *));

 446     if (!ObjDesc->Package.Elements)
 447     {
 448         AcpiUtDeleteObjectDesc (ObjDesc);
 449         return_ACPI_STATUS (AE_NO_MEMORY);
 450     }

 452     ObjDesc->Package.Count = ElementCount;

 454     /*
 455      * Initialize the elements of the package, up to the NumElements count.
 456      * Package is automatically padded with uninitialized (NULL) elements
 457      * if NumElements is greater than the package list length. Likewise,
 458      * Package is truncated if NumElements is less than the list length.
 459      */
 460     Arg = Op->Common.Value.Arg;
 461     Arg = Arg->Common.Next;
 462     for (i = 0; Arg && (i < ElementCount); i++)
 463     {
 464         if (Arg->Common.AmlOpcode == AML_INT_RETURN_VALUE_OP)
 465         {
 466             if (Arg->Common.Node->Type == ACPI_TYPE_METHOD)
 467             {
 468                 /*
 469                  * A method reference "looks" to the parser to be a method
 470                  * invocation, so we special case it here
 471                  */
 472                 Arg->Common.AmlOpcode = AML_INT_NAMEPATH_OP;
 473                 Status = AcpiDsBuildInternalObject (WalkState, Arg,
 474                             &ObjDesc->Package.Elements[i]);
 475             }
 476             else
 477             {
 478                 /* This package element is already built, just get it */

 480                 ObjDesc->Package.Elements[i] =
 481                     ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Arg->Common.Node);
 482             }
 483         }
 484         else
 485         {
 486             Status = AcpiDsBuildInternalObject (WalkState, Arg,
 487                         &ObjDesc->Package.Elements[i]);
 488         }

 490         if (*ObjDescPtr)
 491         {
 492             /* Existing package, get existing reference count */

 494             ReferenceCount = (*ObjDescPtr)->Common.ReferenceCount;
 495             if (ReferenceCount > 1)
 496             {
 497                 /* Make new element ref count match original ref count */

 499                 for (Index = 0; Index < (ReferenceCount - 1); Index++)
 500                 {
 501                     AcpiUtAddReference ((ObjDesc->Package.Elements[i]));
 502                 }
 503             }
 504         }
```

```
506         Arg = Arg->Common.Next;
507     }

509     /* Check for match between NumElements and actual length of PackageList */

511     if (Arg)
512     {
513         /*
514          * NumElements was exhausted, but there are remaining elements in the
515          * PackageList. Truncate the package to NumElements.
516          *
517          * Note: technically, this is an error, from ACPI spec: "It is an error
518          * for NumElements to be less than the number of elements in the
519          * PackageList". However, we just print a message and
520          * no exception is returned. This provides Windows compatibility. Some
521          * BIOSs will alter the NumElements on the fly, creating this type
522          * of ill-formed package object.
523          */
524         while (Arg)
525         {
526             /*
527              * We must delete any package elements that were created earlier
528              * and are not going to be used because of the package truncation.
529              */
530             if (Arg->Common.Node)
531             {
532                 AcpiUtRemoveReference (
533                     ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Arg->Common.Node));
534                 Arg->Common.Node = NULL;
535             }

537             /* Find out how many elements there really are */

539             i++;
540             Arg = Arg->Common.Next;
541         }

543         ACPI_INFO ((AE_INFO,
544             "Actual Package length (%u) is larger than NumElements field (%u), t
544             "Actual Package length (%u) is larger than NumElements field (%u), t
545             i, ElementCount));
546     }
547     else if (i < ElementCount)
548     {
549         /*
550          * Arg list (elements) was exhausted, but we did not reach NumElements c
551          * Note: this is not an error, the package is padded out with NULLs.
552          */
553         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
554             "Package List length (%u) smaller than NumElements count (%u), padde
555             i, ElementCount));
556     }

558     ObjDesc->Package.Flags |= AOPOBJ_DATA_VALID;
559     Op->Common.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjDesc);
560     return_ACPI_STATUS (Status);
561 }
```

*unchanged_portion_omitted_*

```
631 #endif /* ACPI_NO_METHOD_EXECUTION */


634 /*****************************************************************************
635  *
636  * FUNCTION:    AcpiDsInitObjectFromOp
637  *
```



```
638  * PARAMETERS:  WalkState       - Current walk state
639  *              Op              - Parser op used to init the internal object
640  *              Opcode          - AML opcode associated with the object
641  *              RetObjDesc      - Namespace object to be initialized
642  *
643  * RETURN:      Status
644  *
645  * DESCRIPTION: Initialize a namespace object from a parser Op and its
646  *              associated arguments. The namespace object is a more compact
647  *              representation of the Op and its arguments.
648  *
649  ****************************************************************************/

651 ACPI_STATUS
652 AcpiDsInitObjectFromOp (
653     ACPI_WALK_STATE         *WalkState,
654     ACPI_PARSE_OBJECT       *Op,
655     UINT16                  Opcode,
656     ACPI_OPERAND_OBJECT     **RetObjDesc)
657 {
658     const ACPI_OPCODE_INFO  *OpInfo;
659     ACPI_OPERAND_OBJECT     *ObjDesc;
660     ACPI_STATUS             Status = AE_OK;


663     ACPI_FUNCTION_TRACE (DsInitObjectFromOp);


666     ObjDesc = *RetObjDesc;
667     OpInfo = AcpiPsGetOpcodeInfo (Opcode);
668     if (OpInfo->Class == AML_CLASS_UNKNOWN)
669     {
670         /* Unknown opcode */

672         return_ACPI_STATUS (AE_TYPE);
673     }

675     /* Perform per-object initialization */

677     switch (ObjDesc->Common.Type)
678     {
679     case ACPI_TYPE_BUFFER:

680         /*
681          * Defer evaluation of Buffer TermArg operand
682          */
683         ObjDesc->Buffer.Node      = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
684                                         WalkState->Operands[0]);
685         ObjDesc->Buffer.AmlStart  = Op->Named.Data;
686         ObjDesc->Buffer.AmlLength = Op->Named.Length;
687         break;


689     case ACPI_TYPE_PACKAGE:

690         /*
691          * Defer evaluation of Package TermArg operand
692          */
693         ObjDesc->Package.Node      = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
694                                         WalkState->Operands[0]);
695         ObjDesc->Package.AmlStart  = Op->Named.Data;
696         ObjDesc->Package.AmlLength = Op->Named.Length;
697         break;


699     case ACPI_TYPE_INTEGER:
```

```
701         switch (OpInfo->Type)
702         {
703         case AML_TYPE_CONSTANT:
704             /*
705              * Resolve AML Constants here - AND ONLY HERE!
706              * All constants are integers.
707              * We mark the integer with a flag that indicates that it started
708              * life as a constant -- so that stores to constants will perform
709              * as expected (noop). ZeroOp is used as a placeholder for optional
710              * target operands.
711              */
712             ObjDesc->Common.Flags = AOPOBJ_AML_CONSTANT;
713
714             switch (Opcode)
715             {
716             case AML_ZERO_OP:
717
718                 ObjDesc->Integer.Value = 0;
719                 break;
720
721             case AML_ONE_OP:
722
723                 ObjDesc->Integer.Value = 1;
724                 break;
725
726             case AML_ONES_OP:
727
728                 ObjDesc->Integer.Value = ACPI_UINT64_MAX;
729
730                 /* Truncate value if we are executing from a 32-bit ACPI table *
731
732 #ifndef ACPI_NO_METHOD_EXECUTION
733                 (void) AcpiExTruncateFor32bitTable (ObjDesc);
737                 AcpiExTruncateFor32bitTable (ObjDesc);
734 #endif
735                 break;
736
737             case AML_REVISION_OP:
738
739                 ObjDesc->Integer.Value = ACPI_CA_VERSION;
740                 break;
741
742             default:
743
744                 ACPI_ERROR ((AE_INFO,
745                     "Unknown constant opcode 0x%X", Opcode));
746                 Status = AE_AML_OPERAND_TYPE;
747                 break;
748             }
749             break;
750
751         case AML_TYPE_LITERAL:
752
753             ObjDesc->Integer.Value = Op->Common.Value.Integer;
754
755 #ifndef ACPI_NO_METHOD_EXECUTION
756             if (AcpiExTruncateFor32bitTable (ObjDesc))
757             {
758                 /* Warn if we found a 64-bit constant in a 32-bit table */
759
760                 ACPI_WARNING ((AE_INFO,
761                     "Truncated 64-bit constant found in 32-bit table: %8.8X%8.8X
762                     ACPI_FORMAT_UINT64 (Op->Common.Value.Integer),
763                     (UINT32) ObjDesc->Integer.Value));
```



```
764             }
760             AcpiExTruncateFor32bitTable (ObjDesc);
765 #endif
766             break;
767
768         default:
769
765         default:
770             ACPI_ERROR ((AE_INFO, "Unknown Integer type 0x%X",
771                 OpInfo->Type));
772             Status = AE_AML_OPERAND_TYPE;
773             break;
774         }
775         break;
776
777     case ACPI_TYPE_STRING:
778
779         ObjDesc->String.Pointer = Op->Common.Value.String;
780         ObjDesc->String.Length = (UINT32) ACPI_STRLEN (Op->Common.Value.String);
781
782         /*
783          * The string is contained in the ACPI table, don't ever try
784          * to delete it
785          */
786         ObjDesc->Common.Flags |= AOPOBJ_STATIC_POINTER;
787         break;
788
789     case ACPI_TYPE_METHOD:
790         break;
791
792     case ACPI_TYPE_LOCAL_REFERENCE:
793
794         switch (OpInfo->Type)
795         {
796         case AML_TYPE_LOCAL_VARIABLE:
797
798             /* Local ID (0-7) is (AML opcode - base AML_LOCAL_OP) */
799
800             ObjDesc->Reference.Value = ((UINT32) Opcode) - AML_LOCAL_OP;
801             ObjDesc->Reference.Class = ACPI_REFCLASS_LOCAL;
802
803 #ifndef ACPI_NO_METHOD_EXECUTION
804             Status = AcpiDsMethodDataGetNode (ACPI_REFCLASS_LOCAL,
805                         ObjDesc->Reference.Value, WalkState,
806                         ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE,
807                             &ObjDesc->Reference.Object));
808 #endif
809             break;
810
811         case AML_TYPE_METHOD_ARGUMENT:
812
813             /* Arg ID (0-6) is (AML opcode - base AML_ARG_OP) */
814
815             ObjDesc->Reference.Value = ((UINT32) Opcode) - AML_ARG_OP;
816             ObjDesc->Reference.Class = ACPI_REFCLASS_ARG;
817
818 #ifndef ACPI_NO_METHOD_EXECUTION
819             Status = AcpiDsMethodDataGetNode (ACPI_REFCLASS_ARG,
820                         ObjDesc->Reference.Value, WalkState,
821                         ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE,
822                             &ObjDesc->Reference.Object));
823 #endif
```

```
824             break;
826         default: /* Object name or Debug object */
828             switch (Op->Common.AmlOpcode)
829             {
830             case AML_INT_NAMEPATH_OP:
832                 /* Node was saved in Op */
834                 ObjDesc->Reference.Node = Op->Common.Node;
835                 ObjDesc->Reference.Object = Op->Common.Node->Object;
836                 ObjDesc->Reference.Class = ACPI_REFCLASS_NAME;
837                 break;
839             case AML_DEBUG_OP:
841                 ObjDesc->Reference.Class = ACPI_REFCLASS_DEBUG;
842                 break;
844             default:
846                 ACPI_ERROR ((AE_INFO,
847                     "Unimplemented reference type for AML opcode: 0x%4.4X", Opco
848                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
849             }
850             break;
851         }
852         break;
854     default:
856         ACPI_ERROR ((AE_INFO, "Unimplemented data type: 0x%X",
857             ObjDesc->Common.Type));
859         Status = AE_AML_OPERAND_TYPE;
860         break;
861     }
863     return_ACPI_STATUS (Status);
864 }
```

```
**********************************************************
   22489 Thu Dec 26 13:48:52 2013
new/usr/src/common/acpica/components/dispatcher/dsopcode.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: dsopcode - Dispatcher support for regions and fields
 * Module Name: dsopcode - Dispatcher suport for regions and fields
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DSOPCODE_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdispat.h"
#include "acinterp.h"
#include "acnamesp.h"
#include "acevents.h"
#include "actables.h"

#define _COMPONENT          ACPI_DISPATCHER
        ACPI_MODULE_NAME    ("dsopcode")
```



```
/* Local prototypes */

static ACPI_STATUS
AcpiDsInitBufferField (
    UINT16                  AmlOpcode,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     *BufferDesc,
    ACPI_OPERAND_OBJECT     *OffsetDesc,
    ACPI_OPERAND_OBJECT     *LengthDesc,
    ACPI_OPERAND_OBJECT     *ResultDesc);


/******************************************************************************
 *
 * FUNCTION:    AcpiDsInitializeRegion
 *
 * PARAMETERS:  ObjHandle       - Region namespace node
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Front end to EvInitializeRegion
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDsInitializeRegion (
    ACPI_HANDLE             ObjHandle)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_STATUS             Status;


    ObjDesc = AcpiNsGetAttachedObject (ObjHandle);

    /* Namespace is NOT locked */

    Status = AcpiEvInitializeRegion (ObjDesc, FALSE);
    return (Status);
}
______unchanged_portion_omitted_


/******************************************************************************
 *
 * FUNCTION:    AcpiDsEvalTableRegionOperands
 *
 * PARAMETERS:  WalkState       - Current walk
 *              Op              - A valid region Op object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Get region address and length.
 *              Called from AcpiDsExecEndOp during DataTableRegion parse
 *              tree walk.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDsEvalTableRegionOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     **Operand;
    ACPI_NAMESPACE_NODE     *Node;
```

```
517     ACPI_PARSE_OBJECT       *NextOp;
518     UINT32                  TableIndex;
519     ACPI_TABLE_HEADER       *Table;


522     ACPI_FUNCTION_TRACE_PTR (DsEvalTableRegionOperands, Op);


525     /*
526      * This is where we evaluate the Signature string, OemId string,
527      * and OemTableId string of the Data Table Region declaration
526      * This is where we evaluate the SignatureString and OemIDString
527      * and OemTableIDString of the DataTableRegion declaration
528      */
529     Node =  Op->Common.Node;

531     /* NextOp points to Signature string op */
531     /* NextOp points to SignatureString op */

533     NextOp = Op->Common.Value.Arg;

535     /*
536      * Evaluate/create the Signature string, OemId string,
537      * and OemTableId string operands
536      * Evaluate/create the SignatureString and OemIDString
537      * and OemTableIDString operands
538      */
539     Status = AcpiDsCreateOperands (WalkState, NextOp);
540     if (ACPI_FAILURE (Status))
541     {
542         return_ACPI_STATUS (Status);
543     }

545     /*
546      * Resolve the Signature string, OemId string,
547      * and OemTableId string operands
546      * Resolve the SignatureString and OemIDString
547      * and OemTableIDString operands
548      */
549     Status = AcpiExResolveOperands (Op->Common.AmlOpcode,
550                 ACPI_WALK_OPERANDS, WalkState);
551     if (ACPI_FAILURE (Status))
552     {
553         return_ACPI_STATUS (Status);
554     }

556     Operand = &WalkState->Operands[0];

558     /* Find the ACPI table */

560     Status = AcpiTbFindTable (Operand[0]->String.Pointer,
561                 Operand[1]->String.Pointer, Operand[2]->String.Pointer,
562                 &TableIndex);
563     if (ACPI_FAILURE (Status))
564     {
565         return_ACPI_STATUS (Status);
566     }

568     AcpiUtRemoveReference (Operand[0]);
569     AcpiUtRemoveReference (Operand[1]);
570     AcpiUtRemoveReference (Operand[2]);

572     Status = AcpiGetTableByIndex (TableIndex, &Table);
573     if (ACPI_FAILURE (Status))
574     {
575         return_ACPI_STATUS (Status);
```



```
576     }

578     ObjDesc = AcpiNsGetAttachedObject (Node);
579     if (!ObjDesc)
580     {
581         return_ACPI_STATUS (AE_NOT_EXIST);
582     }

584     ObjDesc->Region.Address = (ACPI_PHYSICAL_ADDRESS) ACPI_TO_INTEGER (Table);
585     ObjDesc->Region.Length = Table->Length;

587     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "RgnObj %p Addr %8.8X%8.8X Len %X\n",
588         ObjDesc,
589         ACPI_FORMAT_NATIVE_UINT (ObjDesc->Region.Address),
590         ObjDesc->Region.Length));

592     /* Now the address and length are valid for this opregion */

594     ObjDesc->Region.Flags |= AOPOBJ_DATA_VALID;

596     return_ACPI_STATUS (Status);
597 }


600 /*******************************************************************************
601  *
602  * FUNCTION:    AcpiDsEvalDataObjectOperands
603  *
604  * PARAMETERS:  WalkState       - Current walk
605  *              Op              - A valid DataObject Op object
606  *              ObjDesc         - DataObject
607  *
608  * RETURN:      Status
609  *
610  * DESCRIPTION: Get the operands and complete the following data object types:
611  *              Buffer, Package.
612  *
613  ******************************************************************************/

615 ACPI_STATUS
616 AcpiDsEvalDataObjectOperands (
617     ACPI_WALK_STATE         *WalkState,
618     ACPI_PARSE_OBJECT       *Op,
619     ACPI_OPERAND_OBJECT     *ObjDesc)
620 {
621     ACPI_STATUS             Status;
622     ACPI_OPERAND_OBJECT     *ArgDesc;
623     UINT32                  Length;


626     ACPI_FUNCTION_TRACE (DsEvalDataObjectOperands);


629     /* The first operand (for all of these data objects) is the length */

631     /*
632      * Set proper index into operand stack for AcpiDsObjStackPush
633      * invoked inside AcpiDsCreateOperand.
634      */
635     WalkState->OperandIndex = WalkState->NumOperands;

637     Status = AcpiDsCreateOperand (WalkState, Op->Common.Value.Arg, 1);
638     if (ACPI_FAILURE (Status))
639     {
640         return_ACPI_STATUS (Status);
641     }
```

```
643     Status = AcpiExResolveOperands (WalkState->Opcode,
644                     &(WalkState->Operands [WalkState->NumOperands -1]),
645                     WalkState);
646     if (ACPI_FAILURE (Status))
647     {
648         return_ACPI_STATUS (Status);
649     }

651     /* Extract length operand */

653     ArgDesc = WalkState->Operands [WalkState->NumOperands - 1];
654     Length = (UINT32) ArgDesc->Integer.Value;

656     /* Cleanup for length operand */

658     Status = AcpiDsObjStackPop (1, WalkState);
659     if (ACPI_FAILURE (Status))
660     {
661         return_ACPI_STATUS (Status);
662     }

664     AcpiUtRemoveReference (ArgDesc);

666     /*
667      * Create the actual data object
668      */
669     switch (Op->Common.AmlOpcode)
670     {
671     case AML_BUFFER_OP:

673         Status = AcpiDsBuildInternalBufferObj (WalkState, Op, Length, &ObjDesc);
674         break;

676     case AML_PACKAGE_OP:
677     case AML_VAR_PACKAGE_OP:

679         Status = AcpiDsBuildInternalPackageObj (WalkState, Op, Length, &ObjDesc)
680         break;

682     default:

684         return_ACPI_STATUS (AE_AML_BAD_OPCODE);
685     }

687     if (ACPI_SUCCESS (Status))
688     {
689         /*
690          * Return the object in the WalkState, unless the parent is a package -
691          * in this case, the return object will be stored in the parse tree
692          * for the package.
693          */
694         if ((!Op->Common.Parent) ||
695             ((Op->Common.Parent->Common.AmlOpcode != AML_PACKAGE_OP) &&
696              (Op->Common.Parent->Common.AmlOpcode != AML_VAR_PACKAGE_OP) &&
697              (Op->Common.Parent->Common.AmlOpcode != AML_NAME_OP)))
698         {
699             WalkState->ResultObj = ObjDesc;
700         }
701     }

703     return_ACPI_STATUS (Status);
704 }
```

unchanged_portion_omitted_

```
**********************************************************
   28491 Thu Dec 26 13:48:53 2013
new/usr/src/common/acpica/components/dispatcher/dsutils.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dsutils - Dispatcher utilities
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSUTILS_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acparser.h"
  49 #include "amlcode.h"
  50 #include "acdispat.h"
  51 #include "acinterp.h"
  52 #include "acnamesp.h"
  53 #include "acdebug.h"

  55 #define _COMPONENT          ACPI_DISPATCHER
  56         ACPI_MODULE_NAME    ("dsutils")
```



```
  59 /*******************************************************************************
  60  *
  61  * FUNCTION:    AcpiDsClearImplicitReturn
  62  *
  63  * PARAMETERS:  WalkState           - Current State
  64  *
  65  * RETURN:      None.
  66  *
  67  * DESCRIPTION: Clear and remove a reference on an implicit return value. Used
  68  *              to delete "stale" return values (if enabled, the return value
  69  *              from every operator is saved at least momentarily, in case the
  70  *              parent method exits.)
  71  *
  72  ******************************************************************************/

  74 void
  75 AcpiDsClearImplicitReturn (
  76     ACPI_WALK_STATE         *WalkState)
  77 {
  78     ACPI_FUNCTION_NAME (DsClearImplicitReturn);


  81     /*
  82      * Slack must be enabled for this feature
  83      */
  84     if (!AcpiGbl_EnableInterpreterSlack)
  85     {
  86         return;
  87     }

  89     if (WalkState->ImplicitReturnObj)
  90     {
  91         /*
  92          * Delete any "stale" implicit return. However, in
  93          * complex statements, the implicit return value can be
  94          * bubbled up several levels.
  95          */
  96         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
  97             "Removing reference on stale implicit return obj %p\n",
  98             WalkState->ImplicitReturnObj));

 100         AcpiUtRemoveReference (WalkState->ImplicitReturnObj);
 101         WalkState->ImplicitReturnObj = NULL;
 102     }
 103 }
______unchanged_portion_omitted_


 176 /*******************************************************************************
 177  *
 178  * FUNCTION:    AcpiDsIsResultUsed
 179  *
 180  * PARAMETERS:  Op                  - Current Op
 181  *              WalkState           - Current State
 182  *
 183  * RETURN:      TRUE if result is used, FALSE otherwise
 184  *
 185  * DESCRIPTION: Check if a result object will be used by the parent
 186  *
 187  ******************************************************************************/

 189 BOOLEAN
 190 AcpiDsIsResultUsed (
 191     ACPI_PARSE_OBJECT       *Op,
 192     ACPI_WALK_STATE         *WalkState)
 193 {
```

```
194     const ACPI_OPCODE_INFO  *ParentInfo;
196     ACPI_FUNCTION_TRACE_PTR (DsIsResultUsed, Op);
199     /* Must have both an Op and a Result Object */
201     if (!Op)
202     {
203         ACPI_ERROR ((AE_INFO, "Null Op"));
204         return_UINT8 (TRUE);
205     }
207     /*
208      * We know that this operator is not a
209      * Return() operator (would not come here.) The following code is the
210      * optional support for a so-called "implicit return". Some AML code
211      * assumes that the last value of the method is "implicitly" returned
212      * to the caller. Just save the last result as the return value.
213      * NOTE: this is optional because the ASL language does not actually
214      * support this behavior.
215      */
216     (void) AcpiDsDoImplicitReturn (WalkState->ResultObj, WalkState, TRUE);
218     /*
219      * Now determine if the parent will use the result
220      *
221      * If there is no parent, or the parent is a ScopeOp, we are executing
222      * at the method level. An executing method typically has no parent,
223      * since each method is parsed separately. A method invoked externally
224      * via ExecuteControlMethod has a ScopeOp as the parent.
225      */
226     if ((!Op->Common.Parent) ||
227         (Op->Common.Parent->Common.AmlOpcode == AML_SCOPE_OP))
228     {
229         /* No parent, the return value cannot possibly be used */
231         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
232             "At Method level, result of [%s] not used\n",
233             AcpiPsGetOpcodeName (Op->Common.AmlOpcode)));
234         return_UINT8 (FALSE);
235     }
237     /* Get info on the parent. The RootOp is AML_SCOPE */
239     ParentInfo = AcpiPsGetOpcodeInfo (Op->Common.Parent->Common.AmlOpcode);
240     if (ParentInfo->Class == AML_CLASS_UNKNOWN)
241     {
242         ACPI_ERROR ((AE_INFO,
243             "Unknown parent opcode Op=%p", Op));
244         return_UINT8 (FALSE);
245     }
247     /*
248      * Decide what to do with the result based on the parent. If
249      * the parent opcode will not use the result, delete the object.
250      * Otherwise leave it as is, it will be deleted when it is used
251      * as an operand later.
252      */
253     switch (ParentInfo->Class)
254     {
255     case AML_CLASS_CONTROL:
257         switch (Op->Common.Parent->Common.AmlOpcode)
258         {
259         case AML_RETURN_OP:
```



```
261             /* Never delete the return value associated with a return opcode */
263             goto ResultUsed;
265         case AML_IF_OP:
266         case AML_WHILE_OP:
267             /*
268              * If we are executing the predicate AND this is the predicate op,
269              * we will use the return value
270              */
271             if ((WalkState->ControlState->Common.State == ACPI_CONTROL_PREDICATE
272                 (WalkState->ControlState->Control.PredicateOp == Op))
273             {
274                 goto ResultUsed;
275             }
276             break;
278         default:
280             /* Ignore other control opcodes */
282             break;
283         }
285         /* The general control opcode returns no result */
287         goto ResultNotUsed;
289     case AML_CLASS_CREATE:
290         /*
291          * These opcodes allow TermArg(s) as operands and therefore
292          * the operands can be method calls. The result is used.
293          */
294         goto ResultUsed;
296     case AML_CLASS_NAMED_OBJECT:
298         if ((Op->Common.Parent->Common.AmlOpcode == AML_REGION_OP)       ||
299             (Op->Common.Parent->Common.AmlOpcode == AML_DATA_REGION_OP)  ||
300             (Op->Common.Parent->Common.AmlOpcode == AML_PACKAGE_OP)      ||
301             (Op->Common.Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP)  ||
302             (Op->Common.Parent->Common.AmlOpcode == AML_BUFFER_OP)       ||
303             (Op->Common.Parent->Common.AmlOpcode == AML_INT_EVAL_SUBTREE_OP) ||
304             (Op->Common.Parent->Common.AmlOpcode == AML_BANK_FIELD_OP))
305         {
306             /*
307              * These opcodes allow TermArg(s) as operands and therefore
308              * the operands can be method calls. The result is used.
309              */
310             goto ResultUsed;
311         }
313         goto ResultNotUsed;
315     default:
316         /*
317          * In all other cases. the parent will actually use the return
318          * object, so keep it.
319          */
```

```
320         goto ResultUsed;
321     }


324 ResultUsed:
325     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
326         "Result of [%s] used by Parent [%s] Op=%p\n",
327         AcpiPsGetOpcodeName (Op->Common.AmlOpcode),
328         AcpiPsGetOpcodeName (Op->Common.Parent->Common.AmlOpcode), Op));

330     return_UINT8 (TRUE);


333 ResultNotUsed:
334     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
335         "Result of [%s] not used by Parent [%s] Op=%p\n",
336         AcpiPsGetOpcodeName (Op->Common.AmlOpcode),
337         AcpiPsGetOpcodeName (Op->Common.Parent->Common.AmlOpcode), Op));

339     return_UINT8 (FALSE);
340 }
```

*unchanged_portion_omitted_*

```
731 /*****************************************************************************
732  *
733  * FUNCTION:    AcpiDsCreateOperands
734  *
735  * PARAMETERS:  WalkState           - Current state
736  *              FirstArg            - First argument of a parser argument tree
737  *
738  * RETURN:      Status
739  *
740  * DESCRIPTION: Convert an operator's arguments from a parse tree format to
741  *              namespace objects and place those argument object on the object
742  *              stack in preparation for evaluation by the interpreter.
743  *
744  ****************************************************************************/

746 ACPI_STATUS
747 AcpiDsCreateOperands (
748     ACPI_WALK_STATE         *WalkState,
749     ACPI_PARSE_OBJECT       *FirstArg)
750 {
751     ACPI_STATUS             Status = AE_OK;
752     ACPI_PARSE_OBJECT       *Arg;
753     ACPI_PARSE_OBJECT       *Arguments[ACPI_OBJ_NUM_OPERANDS];
754     UINT32                  ArgCount = 0;
755     UINT32                  Index = WalkState->NumOperands;
756     UINT32                  i;


759     ACPI_FUNCTION_TRACE_PTR (DsCreateOperands, FirstArg);


762     /* Get all arguments in the list */

764     Arg = FirstArg;
765     while (Arg)
766     {
767         if (Index >= ACPI_OBJ_NUM_OPERANDS)
768         {
769             return_ACPI_STATUS (AE_BAD_DATA);
770         }

772         Arguments[Index] = Arg;
```



```
773         WalkState->Operands [Index] = NULL;

775         /* Move on to next argument, if any */

777         Arg = Arg->Common.Next;
778         ArgCount++;
779         Index++;
780     }

782     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
783         "NumOperands %d, ArgCount %d, Index %d\n",
784         WalkState->NumOperands, ArgCount, Index));
786     Index--;

786     /* Create the interpreter arguments, in reverse order */
788     /* It is the appropriate order to get objects from the Result stack */

788     Index--;
789     for (i = 0; i < ArgCount; i++)
790     {
791         Arg = Arguments[Index];

794         /* Force the filling of the operand stack in inverse order */

792         WalkState->OperandIndex = (UINT8) Index;

794         Status = AcpiDsCreateOperand (WalkState, Arg, Index);
795         if (ACPI_FAILURE (Status))
796         {
797             goto Cleanup;
798         }

800         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
801             "Created Arg #%u (%p) %u args total\n",
802             Index, Arg, ArgCount));
803         Index--;

806         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Arg #%u (%p) done, Arg1=%p\n",
807             Index, Arg, FirstArg));
804     }

806     return_ACPI_STATUS (Status);


809 Cleanup:
810     /*
811      * We must undo everything done above; meaning that we must
812      * pop everything off of the operand stack and delete those
813      * objects
814      */
815     AcpiDsObjStackPopAndDelete (ArgCount, WalkState);

817     ACPI_EXCEPTION ((AE_INFO, Status, "While creating Arg %u", Index));
818     return_ACPI_STATUS (Status);
819 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   23264 Thu Dec 26 13:48:53 2013
new/usr/src/common/acpica/components/dispatcher/dswexec.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dswexec - Dispatcher method execution callbacks;
   4  *                        dispatch to interpreter.
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  45 #define __DSWEXEC_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acparser.h"
  50 #include "amlcode.h"
  51 #include "acdispat.h"
  52 #include "acinterp.h"
  53 #include "acnamesp.h"
  54 #include "acdebug.h"


  57 #define _COMPONENT          ACPI_DISPATCHER
  58         ACPI_MODULE_NAME    ("dswexec")
```



```
  60 /*
  61  * Dispatch table for opcode classes
  62  */
  63 static ACPI_EXECUTE_OP      AcpiGbl_OpTypeDispatch [] =
  64 {
  65     AcpiExOpcode_0A_0T_1R,
  66     AcpiExOpcode_1A_0T_0R,
  67     AcpiExOpcode_1A_0T_1R,
  68     AcpiExOpcode_1A_1T_0R,
  69     AcpiExOpcode_1A_1T_1R,
  70     AcpiExOpcode_2A_0T_0R,
  71     AcpiExOpcode_2A_0T_1R,
  72     AcpiExOpcode_2A_1T_1R,
  73     AcpiExOpcode_2A_2T_1R,
  74     AcpiExOpcode_3A_0T_0R,
  75     AcpiExOpcode_3A_1T_1R,
  76     AcpiExOpcode_6A_0T_1R
  77 };


  80 /*****************************************************************************
  81  *
  82  * FUNCTION:    AcpiDsGetPredicateValue
  83  *
  84  * PARAMETERS:  WalkState       - Current state of the parse tree walk
  85  *              ResultObj       - if non-zero, pop result from result stack
  86  *
  87  * RETURN:      Status
  88  *
  89  * DESCRIPTION: Get the result of a predicate evaluation
  90  *
  91  ****************************************************************************/

  93 ACPI_STATUS
  94 AcpiDsGetPredicateValue (
  95     ACPI_WALK_STATE         *WalkState,
  96     ACPI_OPERAND_OBJECT     *ResultObj)
  97 {
  98     ACPI_STATUS             Status = AE_OK;
  99     ACPI_OPERAND_OBJECT     *ObjDesc;
 100     ACPI_OPERAND_OBJECT     *LocalObjDesc = NULL;


 103     ACPI_FUNCTION_TRACE_PTR (DsGetPredicateValue, WalkState);


 106     WalkState->ControlState->Common.State = 0;

 108     if (ResultObj)
 109     {
 110         Status = AcpiDsResultPop (&ObjDesc, WalkState);
 111         if (ACPI_FAILURE (Status))
 112         {
 113             ACPI_EXCEPTION ((AE_INFO, Status,
 114                 "Could not get result from predicate evaluation"));

 116             return_ACPI_STATUS (Status);
 117         }
 118     }
 119     else
 120     {
 121         Status = AcpiDsCreateOperand (WalkState, WalkState->Op, 0);
 122         if (ACPI_FAILURE (Status))
 123         {
 124             return_ACPI_STATUS (Status);
```

```
125         }
127         Status = AcpiExResolveToValue (&WalkState->Operands [0], WalkState);
128         if (ACPI_FAILURE (Status))
129         {
130             return_ACPI_STATUS (Status);
131         }
133         ObjDesc = WalkState->Operands [0];
134     }
136     if (!ObjDesc)
137     {
138         ACPI_ERROR ((AE_INFO,
139             "No predicate ObjDesc=%p State=%p",
140             ObjDesc, WalkState));
142         return_ACPI_STATUS (AE_AML_NO_OPERAND);
143     }
145     /*
146      * Result of predicate evaluation must be an Integer
147      * object. Implicitly convert the argument if necessary.
148      */
149     Status = AcpiExConvertToInteger (ObjDesc, &LocalObjDesc, 16);
150     if (ACPI_FAILURE (Status))
151     {
152         goto Cleanup;
153     }
155     if (LocalObjDesc->Common.Type != ACPI_TYPE_INTEGER)
156     {
157         ACPI_ERROR ((AE_INFO,
158             "Bad predicate (not an integer) ObjDesc=%p State=%p Type=0x%X",
159             ObjDesc, WalkState, ObjDesc->Common.Type));
161         Status = AE_AML_OPERAND_TYPE;
162         goto Cleanup;
163     }
165     /* Truncate the predicate to 32-bits if necessary */
167     (void) AcpiExTruncateFor32bitTable (LocalObjDesc);
167     AcpiExTruncateFor32bitTable (LocalObjDesc);
169     /*
170      * Save the result of the predicate evaluation on
171      * the control stack
172      */
173     if (LocalObjDesc->Integer.Value)
174     {
175         WalkState->ControlState->Common.Value = TRUE;
176     }
177     else
178     {
179         /*
180          * Predicate is FALSE, we will just toss the
181          * rest of the package
182          */
183         WalkState->ControlState->Common.Value = FALSE;
184         Status = AE_CTRL_FALSE;
185     }
187     /* Predicate can be used for an implicit return value */
189     (void) AcpiDsDoImplicitReturn (LocalObjDesc, WalkState, TRUE);
```



```
192 Cleanup:
194     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Completed a predicate eval=%X Op=%p\n",
195         WalkState->ControlState->Common.Value, WalkState->Op));
197      /* Break to debugger to display result */
199     ACPI_DEBUGGER_EXEC (AcpiDbDisplayResultObject (LocalObjDesc, WalkState));
201     /*
202      * Delete the predicate result object (we know that
203      * we don't need it anymore)
204      */
205     if (LocalObjDesc != ObjDesc)
206     {
207         AcpiUtRemoveReference (LocalObjDesc);
208     }
209     AcpiUtRemoveReference (ObjDesc);
211     WalkState->ControlState->Common.State = ACPI_CONTROL_NORMAL;
212     return_ACPI_STATUS (Status);
213 }
216 /*****************************************************************************
217  *
218  * FUNCTION:    AcpiDsExecBeginOp
219  *
220  * PARAMETERS:  WalkState       - Current state of the parse tree walk
221  *              OutOp           - Where to return op if a new one is created
222  *
223  * RETURN:      Status
224  *
225  * DESCRIPTION: Descending callback used during the execution of control
226  *              methods. This is where most operators and operands are
227  *              dispatched to the interpreter.
228  *
229  ****************************************************************************/
231 ACPI_STATUS
232 AcpiDsExecBeginOp (
233     ACPI_WALK_STATE         *WalkState,
234     ACPI_PARSE_OBJECT       **OutOp)
235 {
236     ACPI_PARSE_OBJECT       *Op;
237     ACPI_STATUS             Status = AE_OK;
238     UINT32                  OpcodeClass;
241     ACPI_FUNCTION_TRACE_PTR (DsExecBeginOp, WalkState);
244     Op = WalkState->Op;
245     if (!Op)
246     {
247         Status = AcpiDsLoad2BeginOp (WalkState, OutOp);
248         if (ACPI_FAILURE (Status))
249         {
250             goto ErrorExit;
251         }
253         Op = *OutOp;
254         WalkState->Op = Op;
255         WalkState->Opcode = Op->Common.AmlOpcode;
```

```
256         WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

258         if (AcpiNsOpensScope (WalkState->OpInfo->ObjectType))
259         {
260             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
261                 "(%s) Popping scope for Op %p\n",
262                 AcpiUtGetTypeName (WalkState->OpInfo->ObjectType), Op));

264             Status = AcpiDsScopeStackPop (WalkState);
265             if (ACPI_FAILURE (Status))
266             {
267                 goto ErrorExit;
268             }
269         }
270     }

272     if (Op == WalkState->Origin)
273     {
274         if (OutOp)
275         {
276             *OutOp = Op;
277         }

279         return_ACPI_STATUS (AE_OK);
280     }

282     /*
283      * If the previous opcode was a conditional, this opcode
284      * must be the beginning of the associated predicate.
285      * Save this knowledge in the current scope descriptor
286      */
287     if ((WalkState->ControlState) &&
288         (WalkState->ControlState->Common.State ==
289             ACPI_CONTROL_CONDITIONAL_EXECUTING))
290     {
291         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Exec predicate Op=%p State=%p\n",
292                         Op, WalkState));

294         WalkState->ControlState->Common.State = ACPI_CONTROL_PREDICATE_EXECUTING

296         /* Save start of predicate */

298         WalkState->ControlState->Control.PredicateOp = Op;
299     }


302     OpcodeClass = WalkState->OpInfo->Class;

304     /* We want to send namepaths to the load code */

306     if (Op->Common.AmlOpcode == AML_INT_NAMEPATH_OP)
307     {
308         OpcodeClass = AML_CLASS_NAMED_OBJECT;
309     }

311     /*
312      * Handle the opcode based upon the opcode type
313      */
314     switch (OpcodeClass)
315     {
316     case AML_CLASS_CONTROL:

318         Status = AcpiDsExecBeginControlOp (WalkState, Op);
319         break;
```



```
321     case AML_CLASS_NAMED_OBJECT:

323         if (WalkState->WalkType & ACPI_WALK_METHOD)
324         {
325             /*
326              * Found a named object declaration during method execution;
327              * we must enter this object into the namespace. The created
328              * object is temporary and will be deleted upon completion of
329              * the execution of this method.
330              *
331              * Note 10/2010: Except for the Scope() op. This opcode does
332              * not actually create a new object, it refers to an existing
333              * object. However, for Scope(), we want to indeed open a
334              * new scope.
335              */
336             if (Op->Common.AmlOpcode != AML_SCOPE_OP)
337             {
338                 Status = AcpiDsLoad2BeginOp (WalkState, NULL);
339             }
340             else
341             {
342                 Status = AcpiDsScopeStackPush (Op->Named.Node,
343                             Op->Named.Node->Type, WalkState);
344                 if (ACPI_FAILURE (Status))
345                 {
346                     return_ACPI_STATUS (Status);
347                 }
348             }
349         }
350         break;


352     case AML_CLASS_EXECUTE:
353     case AML_CLASS_CREATE:

355         break;

357     default:

360     default:
359         break;
360     }

362     /* Nothing to do here during method execution */

364     return_ACPI_STATUS (Status);


367 ErrorExit:
368     Status = AcpiDsMethodError (Status, WalkState);
369     return_ACPI_STATUS (Status);
370 }


373 /*****************************************************************************
374  *
375  * FUNCTION:    AcpiDsExecEndOp
376  *
377  * PARAMETERS:  WalkState       - Current state of the parse tree walk
378  *
379  * RETURN:      Status
380  *
381  * DESCRIPTION: Ascending callback used during the execution of control
382  *              methods. The only thing we really need to do here is to
383  *              notice the beginning of IF, ELSE, and WHILE blocks.
384  *
```

```
385  *****************************************************************************/

387 ACPI_STATUS
388 AcpiDsExecEndOp (
389     ACPI_WALK_STATE         *WalkState)
390 {
391     ACPI_PARSE_OBJECT       *Op;
392     ACPI_STATUS             Status = AE_OK;
393     UINT32                  OpType;
394     UINT32                  OpClass;
395     ACPI_PARSE_OBJECT       *NextOp;
396     ACPI_PARSE_OBJECT       *FirstArg;


399     ACPI_FUNCTION_TRACE_PTR (DsExecEndOp, WalkState);


402     Op      = WalkState->Op;
403     OpType  = WalkState->OpInfo->Type;
404     OpClass = WalkState->OpInfo->Class;

406     if (OpClass == AML_CLASS_UNKNOWN)
407     {
408         ACPI_ERROR ((AE_INFO, "Unknown opcode 0x%X", Op->Common.AmlOpcode));
409         return_ACPI_STATUS (AE_NOT_IMPLEMENTED);
410     }

412     FirstArg = Op->Common.Value.Arg;

414     /* Init the walk state */

416     WalkState->NumOperands = 0;
417     WalkState->OperandIndex = 0;
418     WalkState->ReturnDesc = NULL;
419     WalkState->ResultObj = NULL;

421     /* Call debugger for single step support (DEBUG build only) */

423     ACPI_DEBUGGER_EXEC (Status = AcpiDbSingleStep (WalkState, Op, OpClass));
424     ACPI_DEBUGGER_EXEC (if (ACPI_FAILURE (Status)) {return_ACPI_STATUS (Status);

426     /* Decode the Opcode Class */

428     switch (OpClass)
429     {
430     case AML_CLASS_ARGUMENT:    /* Constants, literals, etc. */

432         if (WalkState->Opcode == AML_INT_NAMEPATH_OP)
433         {
434             Status = AcpiDsEvaluateNamePath (WalkState);
435             if (ACPI_FAILURE (Status))
436             {
437                 goto Cleanup;
438             }
439         }
440         break;


442     case AML_CLASS_EXECUTE:     /* Most operators with arguments */

444         /* Build resolved operand stack */

446         Status = AcpiDsCreateOperands (WalkState, FirstArg);
447         if (ACPI_FAILURE (Status))
448         {
449             goto Cleanup;
```



```
450         }

452         /*
453          * All opcodes require operand resolution, with the only exceptions
454          * being the ObjectType and SizeOf operators.
455          */
456         if (!(WalkState->OpInfo->Flags & AML_NO_OPERAND_RESOLVE))
457         {
458             /* Resolve all operands */

460             Status = AcpiExResolveOperands (WalkState->Opcode,
461                         &(WalkState->Operands [WalkState->NumOperands -1]),
462                         WalkState);
463         }

465         if (ACPI_SUCCESS (Status))
466         {
467             /*
468              * Dispatch the request to the appropriate interpreter handler
469              * routine. There is one routine per opcode "type" based upon the
470              * number of opcode arguments and return type.
471              */
472             Status = AcpiGbl_OpTypeDispatch[OpType] (WalkState);
473         }
474         else
475         {
476             /*
477              * Treat constructs of the form "Store(LocalX,LocalX)" as noops when
478              * Local is uninitialized.
479              */
480             if  ((Status == AE_AML_UNINITIALIZED_LOCAL) &&
481                 (WalkState->Opcode == AML_STORE_OP) &&
482                 (WalkState->Operands[0]->Common.Type == ACPI_TYPE_LOCAL_REFERENC
483                 (WalkState->Operands[1]->Common.Type == ACPI_TYPE_LOCAL_REFERENC
484                 (WalkState->Operands[0]->Reference.Class ==
485                  WalkState->Operands[1]->Reference.Class) &&
486                 (WalkState->Operands[0]->Reference.Value ==
487                  WalkState->Operands[1]->Reference.Value))
488             {
489                 Status = AE_OK;
490             }
491             else
492             {
493                 ACPI_EXCEPTION ((AE_INFO, Status,
494                     "While resolving operands for [%s]",
495                     AcpiPsGetOpcodeName (WalkState->Opcode)));
496             }
497         }

499         /* Always delete the argument objects and clear the operand stack */

501         AcpiDsClearOperands (WalkState);

503         /*
504          * If a result object was returned from above, push it on the
505          * current result stack
506          */
507         if (ACPI_SUCCESS (Status) &&
508             WalkState->ResultObj)
509         {
510             Status = AcpiDsResultPush (WalkState->ResultObj, WalkState);
511         }
512         break;


514     default:
```

```
516         switch (OpType)
517         {
518         case AML_TYPE_CONTROL:    /* Type 1 opcode, IF/ELSE/WHILE/NOOP */

520             /* 1 Operand, 0 ExternalResult, 0 InternalResult */

522             Status = AcpiDsExecEndControlOp (WalkState, Op);

524             break;


526         case AML_TYPE_METHOD_CALL:

527             /*
528              * If the method is referenced from within a package
529              * declaration, it is not a invocation of the method, just
530              * a reference to it.
531              */
532             if ((Op->Asl.Parent) &&
533                ((Op->Asl.Parent->Asl.AmlOpcode == AML_PACKAGE_OP) ||
534                 (Op->Asl.Parent->Asl.AmlOpcode == AML_VAR_PACKAGE_OP)))
535             {
536                 ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
537                     "Method Reference in a Package, Op=%p\n", Op));

539                 Op->Common.Node = (ACPI_NAMESPACE_NODE *) Op->Asl.Value.Arg->Asl
540                 AcpiUtAddReference (Op->Asl.Value.Arg->Asl.Node->Object);
541                 return_ACPI_STATUS (AE_OK);
542             }

544             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
545                 "Method invocation, Op=%p\n", Op));
550             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Method invocation, Op=%p\n", O

547             /*
548              * (AML_METHODCALL) Op->Asl.Value.Arg->Asl.Node contains
549              * the method Node pointer
550              */
551             /* NextOp points to the op that holds the method name */

553             NextOp = FirstArg;

555             /* NextOp points to first argument op */

557             NextOp = NextOp->Common.Next;

559             /*
560              * Get the method's arguments and put them on the operand stack
561              */
562             Status = AcpiDsCreateOperands (WalkState, NextOp);
563             if (ACPI_FAILURE (Status))
564             {
565                 break;
566             }

568             /*
569              * Since the operands will be passed to another control method,
570              * we must resolve all local references here (Local variables,
571              * arguments to *this* method, etc.)
572              */
573             Status = AcpiDsResolveOperands (WalkState);
574             if (ACPI_FAILURE (Status))
575             {
576                 /* On error, clear all resolved operands */
```



```
578                 AcpiDsClearOperands (WalkState);
579                 break;
580             }

582             /*
583              * Tell the walk loop to preempt this running method and
584              * execute the new method
585              */
586             Status = AE_CTRL_TRANSFER;

588             /*
589              * Return now; we don't want to disturb anything,
590              * especially the operand count!
591              */
592             return_ACPI_STATUS (Status);


594         case AML_TYPE_CREATE_FIELD:

596             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
597                 "Executing CreateField Buffer/Index Op=%p\n", Op));

599             Status = AcpiDsLoad2EndOp (WalkState);
600             if (ACPI_FAILURE (Status))
601             {
602                 break;
603             }

605             Status = AcpiDsEvalBufferFieldOperands (WalkState, Op);
606             break;


609         case AML_TYPE_CREATE_OBJECT:

611             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
612                 "Executing CreateObject (Buffer/Package) Op=%p\n", Op));

614             switch (Op->Common.Parent->Common.AmlOpcode)
615             {
616             case AML_NAME_OP:

617                 /*
618                  * Put the Node on the object stack (Contains the ACPI Name
619                  * of this object)
620                  */
621                 WalkState->Operands[0] = (void *) Op->Common.Parent->Common.Node
622                 WalkState->NumOperands = 1;

624                 Status = AcpiDsCreateNode (WalkState,
625                             Op->Common.Parent->Common.Node,
626                             Op->Common.Parent);
627                 if (ACPI_FAILURE (Status))
628                 {
629                     break;
630                 }

632                 /* Fall through */
633                 /*lint -fallthrough */

635             case AML_INT_EVAL_SUBTREE_OP:

637                 Status = AcpiDsEvalDataObjectOperands (WalkState, Op,
638                             AcpiNsGetAttachedObject (Op->Common.Parent->Common.N
639                 break;

641             default:
```

```
643                 Status = AcpiDsEvalDataObjectOperands (WalkState, Op, NULL);
644                 break;
645             }

647             /*
648              * If a result object was returned from above, push it on the
649              * current result stack
650              */
651             if (WalkState->ResultObj)
652             {
653                 Status = AcpiDsResultPush (WalkState->ResultObj, WalkState);
654             }
655             break;


657         case AML_TYPE_NAMED_FIELD:
658         case AML_TYPE_NAMED_COMPLEX:
659         case AML_TYPE_NAMED_SIMPLE:
660         case AML_TYPE_NAMED_NO_OBJ:

662             Status = AcpiDsLoad2EndOp (WalkState);
663             if (ACPI_FAILURE (Status))
664             {
665                 break;
666             }

668             if (Op->Common.AmlOpcode == AML_REGION_OP)
669             {
670                 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
671                     "Executing OpRegion Address/Length Op=%p\n", Op));

673                 Status = AcpiDsEvalRegionOperands (WalkState, Op);
674                 if (ACPI_FAILURE (Status))
675                 {
676                     break;
677                 }
678             }
679             else if (Op->Common.AmlOpcode == AML_DATA_REGION_OP)
680             {
681                 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
682                     "Executing DataTableRegion Strings Op=%p\n", Op));

684                 Status = AcpiDsEvalTableRegionOperands (WalkState, Op);
685                 if (ACPI_FAILURE (Status))
686                 {
687                     break;
688                 }
689             }
690             else if (Op->Common.AmlOpcode == AML_BANK_FIELD_OP)
691             {
692                 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
693                     "Executing BankField Op=%p\n", Op));

695                 Status = AcpiDsEvalBankFieldOperands (WalkState, Op);
696                 if (ACPI_FAILURE (Status))
697                 {
698                     break;
699                 }
700             }
701             break;


703         case AML_TYPE_UNDEFINED:

705             ACPI_ERROR ((AE_INFO,
```



```
706                 "Undefined opcode type Op=%p", Op));
707             return_ACPI_STATUS (AE_NOT_IMPLEMENTED);


709         case AML_TYPE_BOGUS:

711             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
712                 "Internal opcode=%X type Op=%p\n",
713                 WalkState->Opcode, Op));
714             break;


716         default:

718             ACPI_ERROR ((AE_INFO,
719                 "Unimplemented opcode, class=0x%X type=0x%X Opcode=0x%X Op=%p",
730                 "Unimplemented opcode, class=0x%X type=0x%X Opcode=-0x%X Op=%p",
720                 OpClass, OpType, Op->Common.AmlOpcode, Op));

722             Status = AE_NOT_IMPLEMENTED;
723             break;
724         }
725     }

727     /*
728      * ACPI 2.0 support for 64-bit integers: Truncate numeric
729      * result value if we are executing from a 32-bit ACPI table
730      */
731     (void) AcpiExTruncateFor32bitTable (WalkState->ResultObj);
742     AcpiExTruncateFor32bitTable (WalkState->ResultObj);

733     /*
734      * Check if we just completed the evaluation of a
735      * conditional predicate
736      */
737     if ((ACPI_SUCCESS (Status)) &&
738         (WalkState->ControlState) &&
739         (WalkState->ControlState->Common.State ==
740             ACPI_CONTROL_PREDICATE_EXECUTING) &&
741         (WalkState->ControlState->Control.PredicateOp == Op))
742     {
743         Status = AcpiDsGetPredicateValue (WalkState, WalkState->ResultObj);
744         WalkState->ResultObj = NULL;
745     }


748 Cleanup:

750     if (WalkState->ResultObj)
751     {
752         /* Break to debugger to display result */

754         ACPI_DEBUGGER_EXEC (AcpiDbDisplayResultObject (WalkState->ResultObj,
755                                 WalkState));

757         /*
758          * Delete the result op if and only if:
759          * Parent will not use the result -- such as any
760          * non-nested type2 op in a method (parent will be method)
761          */
762         AcpiDsDeleteResultIfNotUsed (Op, WalkState->ResultObj, WalkState);
763     }

765 #ifdef _UNDER_DEVELOPMENT

767     if (WalkState->ParserState.Aml == WalkState->ParserState.AmlEnd)
```

```
768     {
769         AcpiDbMethodEnd (WalkState);
770     }
771 #endif

773     /* Invoke exception handler on error */

775     if (ACPI_FAILURE (Status))
776     {
777         Status = AcpiDsMethodError (Status, WalkState);
778     }

780     /* Always clear the object stack */

782     WalkState->NumOperands = 0;
783     return_ACPI_STATUS (Status);
784 }
```

```
**********************************************************
   17940 Thu Dec 26 13:48:53 2013
new/usr/src/common/acpica/components/dispatcher/dswload.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dswload - Dispatcher first pass namespace load callbacks
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __DSWLOAD_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acparser.h"
  49 #include "amlcode.h"
  50 #include "acdispat.h"
  51 #include "acinterp.h"
  52 #include "acnamesp.h"

  54 #ifdef ACPI_ASL_COMPILER
  55 #include "acdisasm.h"
  56 #endif

  58 #define _COMPONENT          ACPI_DISPATCHER
```



```
  59         ACPI_MODULE_NAME    ("dswload")


  62 /*****************************************************************************
  63  *
  64  * FUNCTION:    AcpiDsInitCallbacks
  65  *
  66  * PARAMETERS:  WalkState       - Current state of the parse tree walk
  67  *              PassNumber      - 1, 2, or 3
  68  *
  69  * RETURN:      Status
  70  *
  71  * DESCRIPTION: Init walk state callbacks
  72  *
  73  ****************************************************************************/

  75 ACPI_STATUS
  76 AcpiDsInitCallbacks (
  77     ACPI_WALK_STATE         *WalkState,
  78     UINT32                  PassNumber)
  79 {

  81     switch (PassNumber)
  82     {
  83     case 1:

  85         WalkState->ParseFlags         = ACPI_PARSE_LOAD_PASS1 |
  86                                         ACPI_PARSE_DELETE_TREE;
  87         WalkState->DescendingCallback = AcpiDsLoad1BeginOp;
  88         WalkState->AscendingCallback  = AcpiDsLoad1EndOp;
  89         break;

  91     case 2:

  93         WalkState->ParseFlags         = ACPI_PARSE_LOAD_PASS1 |
  94                                         ACPI_PARSE_DELETE_TREE;
  95         WalkState->DescendingCallback = AcpiDsLoad2BeginOp;
  96         WalkState->AscendingCallback  = AcpiDsLoad2EndOp;
  97         break;

  99     case 3:

 101 #ifndef ACPI_NO_METHOD_EXECUTION
 102         WalkState->ParseFlags        |= ACPI_PARSE_EXECUTE  |
 103                                         ACPI_PARSE_DELETE_TREE;
 104         WalkState->DescendingCallback = AcpiDsExecBeginOp;
 105         WalkState->AscendingCallback  = AcpiDsExecEndOp;
 106 #endif
 107         break;

 109     default:

 111         return (AE_BAD_PARAMETER);
 112     }

 114     return (AE_OK);
 115 }


 118 /*****************************************************************************
 119  *
 120  * FUNCTION:    AcpiDsLoad1BeginOp
 121  *
 122  * PARAMETERS:  WalkState       - Current state of the parse tree walk
 123  *              OutOp           - Where to return op if a new one is created
 124  *
```

```
 * RETURN:      Status
 *
 * DESCRIPTION: Descending callback used during the loading of ACPI tables.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDsLoad1BeginOp (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       **OutOp)
{
    ACPI_PARSE_OBJECT       *Op;
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;
    ACPI_OBJECT_TYPE        ObjectType;
    char                    *Path;
    UINT32                  Flags;


    ACPI_FUNCTION_TRACE (DsLoad1BeginOp);


    Op = WalkState->Op;
    ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p State=%p\n", Op, WalkState));

    /* We are only interested in opcodes that have an associated name */

    if (Op)
    {
        if (!(WalkState->OpInfo->Flags & AML_NAMED))
        {
            *OutOp = Op;
            return_ACPI_STATUS (AE_OK);
        }

        /* Check if this object has already been installed in the namespace */

        if (Op->Common.Node)
        {
            *OutOp = Op;
            return_ACPI_STATUS (AE_OK);
        }
    }

    Path = AcpiPsGetNextNamestring (&WalkState->ParserState);

    /* Map the raw opcode into an internal object type */

    ObjectType = WalkState->OpInfo->ObjectType;

    ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
        "State=%p Op=%p [%s]\n", WalkState, Op, AcpiUtGetTypeName (ObjectType)));

    switch (WalkState->Opcode)
    {
    case AML_SCOPE_OP:

        /*
         * The target name of the Scope() operator must exist at this point so
         * that we can actually open the scope to enter new names underneath it.
         * Allow search-to-root for single namesegs.
         */
        Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
                    ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT, WalkState, &(
#ifdef ACPI_ASL_COMPILER
        if (Status == AE_NOT_FOUND)
```



```
        {
            /*
             * Table disassembly:
             * Target of Scope() not found. Generate an External for it, and
             * insert the name into the namespace.
             */
            AcpiDmAddOpToExternalList (Op, Path, ACPI_TYPE_DEVICE, 0, 0);
            AcpiDmAddToExternalList (Op, Path, ACPI_TYPE_DEVICE, 0);
            Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
                        ACPI_IMODE_LOAD_PASS1, ACPI_NS_SEARCH_PARENT,
                        WalkState, &Node);
        }
#endif
        if (ACPI_FAILURE (Status))
        {
            ACPI_ERROR_NAMESPACE (Path, Status);
            return_ACPI_STATUS (Status);
        }

        /*
         * Check to make sure that the target is
         * one of the opcodes that actually opens a scope
         */
        switch (Node->Type)
        {
        case ACPI_TYPE_ANY:
        case ACPI_TYPE_LOCAL_SCOPE:         /* Scope  */
        case ACPI_TYPE_DEVICE:
        case ACPI_TYPE_POWER:
        case ACPI_TYPE_PROCESSOR:
        case ACPI_TYPE_THERMAL:

            /* These are acceptable types */
            break;

        case ACPI_TYPE_INTEGER:
        case ACPI_TYPE_STRING:
        case ACPI_TYPE_BUFFER:

            /*
             * These types we will allow, but we will change the type.
             * This enables some existing code of the form:
             *
             *  Name (DEB, 0)
             *  Scope (DEB) { ... }
             *
             * Note: silently change the type here. On the second pass,
             * we will report a warning
             */
            ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
                "Type override - [%4.4s] had invalid type (%s) "
                "for Scope operator, changed to type ANY\n",
                AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type)));

            Node->Type = ACPI_TYPE_ANY;
            WalkState->ScopeInfo->Common.Value = ACPI_TYPE_ANY;
            break;

        case ACPI_TYPE_METHOD:
            /*
             * Allow scope change to root during execution of module-level
             * code. Root is typed METHOD during this time.
             */
            if ((Node == AcpiGbl_RootNode) &&
                (WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
            {
```

```
254                 break;
255             }

257             /*lint -fallthrough */

259         default:

261             /* All other types are an error */

263             ACPI_ERROR ((AE_INFO,
264                 "Invalid type (%s) for target of "
265                 "Scope operator [%4.4s] (Cannot override)",
266                 AcpiUtGetTypeName (Node->Type), AcpiUtGetNodeName (Node)));

268             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
269         }
270         break;


272     default:
273         /*
274          * For all other named opcodes, we will enter the name into
275          * the namespace.
276          *
277          * Setup the search flags.
278          * Since we are entering a name into the namespace, we do not want to
279          * enable the search-to-root upsearch.
280          *
281          * There are only two conditions where it is acceptable that the name
282          * already exists:
283          *    1) the Scope() operator can reopen a scoping object that was
284          *       previously defined (Scope, Method, Device, etc.)
285          *    2) Whenever we are parsing a deferred opcode (OpRegion, Buffer,
286          *       BufferField, or Package), the name of the object is already
287          *       in the namespace.
288          */
289         if (WalkState->DeferredNode)
290         {
291             /* This name is already in the namespace, get the node */

293             Node = WalkState->DeferredNode;
294             Status = AE_OK;
295             break;
296         }

298         /*
299          * If we are executing a method, do not create any namespace objects
300          * during the load phase, only during execution.
301          */
302         if (WalkState->MethodNode)
303         {
304             Node = NULL;
305             Status = AE_OK;
306             break;
307         }

309         Flags = ACPI_NS_NO_UPSEARCH;
310         if ((WalkState->Opcode != AML_SCOPE_OP) &&
311             (!(WalkState->ParseFlags & ACPI_PARSE_DEFERRED_OP)))
312         {
313             Flags |= ACPI_NS_ERROR_IF_FOUND;
314             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "[%s] Cannot already exist\n",
315                     AcpiUtGetTypeName (ObjectType)));
316         }
317         else
318         {
```



```
319             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
320                 "[%s] Both Find or Create allowed\n",
321                     AcpiUtGetTypeName (ObjectType)));
322         }

324         /*
325          * Enter the named type into the internal namespace. We enter the name
326          * as we go downward in the parse tree. Any necessary subobjects that
327          * involve arguments to the opcode must be created as we go back up the
328          * parse tree later.
329          */
330         Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
331                     ACPI_IMODE_LOAD_PASS1, Flags, WalkState, &Node);
332         if (ACPI_FAILURE (Status))
333         {
334             if (Status == AE_ALREADY_EXISTS)
335             {
336                 /* The name already exists in this scope */

338                 if (Node->Flags & ANOBJ_IS_EXTERNAL)
339                 {
340                     /*
341                      * Allow one create on an object or segment that was
342                      * previously declared External
343                      */
344                     Node->Flags &= ~ANOBJ_IS_EXTERNAL;
345                     Node->Type = (UINT8) ObjectType;

347                     /* Just retyped a node, probably will need to open a scope *

349                     if (AcpiNsOpensScope (ObjectType))
350                     {
351                         Status = AcpiDsScopeStackPush (Node, ObjectType, WalkSta
352                         if (ACPI_FAILURE (Status))
353                         {
354                             return_ACPI_STATUS (Status);
355                         }
356                     }

358                     Status = AE_OK;
359                 }
360             }

362             if (ACPI_FAILURE (Status))
363             {
364                 ACPI_ERROR_NAMESPACE (Path, Status);
365                 return_ACPI_STATUS (Status);
366             }
367         }
368         break;
369     }

371     /* Common exit */

373     if (!Op)
374     {
375         /* Create a new op */

377         Op = AcpiPsAllocOp (WalkState->Opcode);
378         if (!Op)
379         {
380             return_ACPI_STATUS (AE_NO_MEMORY);
381         }
382     }

384     /* Initialize the op */
```

```
#if (defined (ACPI_NO_METHOD_EXECUTION) || defined (ACPI_CONSTANT_EVAL_ONLY))
    Op->Named.Path = ACPI_CAST_PTR (UINT8, Path);
#endif

    if (Node)
    {
        /*
         * Put the Node in the "op" object that the parser uses, so we
         * can get it again quickly when this scope is closed
         */
        Op->Common.Node = Node;
        Op->Named.Name = Node->Name.Integer;
    }

    AcpiPsAppendArg (AcpiPsGetParentScope (&WalkState->ParserState), Op);
    *OutOp = Op;
    return_ACPI_STATUS (Status);
}
```

unchanged_portion_omitted_

****
21930 Thu Dec 26 13:48:54 2013
new/usr/src/common/acpica/components/dispatcher/dswload2.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dswload2 - Dispatcher second pass namespace load callbacks
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DSWLOAD2_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acdispat.h"
#include "acinterp.h"
#include "acnamesp.h"
#include "acevents.h"

#define _COMPONENT          ACPI_DISPATCHER
        ACPI_MODULE_NAME    ("dswload2")


/*****************************************************************************
```



```
 *
 * FUNCTION:    AcpiDsLoad2BeginOp
 *
 * PARAMETERS:  WalkState       - Current state of the parse tree walk
 *              OutOp           - Wher to return op if a new one is created
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Descending callback used during the loading of ACPI tables.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDsLoad2BeginOp (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       **OutOp)
{
    ACPI_PARSE_OBJECT       *Op;
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;
    ACPI_OBJECT_TYPE        ObjectType;
    char                    *BufferPtr;
    UINT32                  Flags;


    ACPI_FUNCTION_TRACE (DsLoad2BeginOp);


    Op = WalkState->Op;
    ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p State=%p\n", Op, WalkState));

    if (Op)
    {
        if ((WalkState->ControlState) &&
            (WalkState->ControlState->Common.State ==
                ACPI_CONTROL_CONDITIONAL_EXECUTING))
        {
            /* We are executing a while loop outside of a method */

            Status = AcpiDsExecBeginOp (WalkState, OutOp);
            return_ACPI_STATUS (Status);
        }

        /* We only care about Namespace opcodes here */

        if ((!(WalkState->OpInfo->Flags & AML_NSOPCODE)   &&
              (WalkState->Opcode != AML_INT_NAMEPATH_OP)) ||
            (!(WalkState->OpInfo->Flags & AML_NAMED)))
        {
            return_ACPI_STATUS (AE_OK);
        }

        /* Get the name we are going to enter or lookup in the namespace */

        if (WalkState->Opcode == AML_INT_NAMEPATH_OP)
        {
            /* For Namepath op, get the path string */

            BufferPtr = Op->Common.Value.String;
            if (!BufferPtr)
            {
                /* No name, just exit */

                return_ACPI_STATUS (AE_OK);
            }
        }
```

```
126         else
127         {
128             /* Get name from the op */

130             BufferPtr = ACPI_CAST_PTR (char, &Op->Named.Name);
131         }
132     }
133     else
134     {
135         /* Get the namestring from the raw AML */

137         BufferPtr = AcpiPsGetNextNamestring (&WalkState->ParserState);
138     }

140     /* Map the opcode into an internal object type */

142     ObjectType = WalkState->OpInfo->ObjectType;

144     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
145         "State=%p Op=%p Type=%X\n", WalkState, Op, ObjectType));

147     switch (WalkState->Opcode)
148     {
149     case AML_FIELD_OP:
150     case AML_BANK_FIELD_OP:
151     case AML_INDEX_FIELD_OP:

153         Node = NULL;
154         Status = AE_OK;
155         break;

157     case AML_INT_NAMEPATH_OP:
158         /*
159          * The NamePath is an object reference to an existing object.
160          * Don't enter the name into the namespace, but look it up
161          * for use later.
162          */
163         Status = AcpiNsLookup (WalkState->ScopeInfo, BufferPtr, ObjectType,
164                         ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT,
165                         WalkState, &(Node));
166         break;

168     case AML_SCOPE_OP:

170         /* Special case for Scope(\) -> refers to the Root node */

172         if (Op && (Op->Named.Node == AcpiGbl_RootNode))
173         {
174             Node = Op->Named.Node;

176             Status = AcpiDsScopeStackPush (Node, ObjectType, WalkState);
177             if (ACPI_FAILURE (Status))
178             {
179                 return_ACPI_STATUS (Status);
180             }
181         }
182         else
183         {
184             /*
185              * The Path is an object reference to an existing object.
186              * Don't enter the name into the namespace, but look it up
187              * for use later.
188              */
189             Status = AcpiNsLookup (WalkState->ScopeInfo, BufferPtr, ObjectType,
190                         ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT,
191                         WalkState, &(Node));
```



```
192             if (ACPI_FAILURE (Status))
193             {
194 #ifdef ACPI_ASL_COMPILER
195                 if (Status == AE_NOT_FOUND)
196                 {
197                     Status = AE_OK;
198                 }
199                 else
200                 {
201                     ACPI_ERROR_NAMESPACE (BufferPtr, Status);
202                 }
203 #else
204                 ACPI_ERROR_NAMESPACE (BufferPtr, Status);
205 #endif
206                 return_ACPI_STATUS (Status);
207             }
208         }

210         /*
211          * We must check to make sure that the target is
212          * one of the opcodes that actually opens a scope
213          */
214         switch (Node->Type)
215         {
216         case ACPI_TYPE_ANY:
217         case ACPI_TYPE_LOCAL_SCOPE:         /* Scope */
218         case ACPI_TYPE_DEVICE:
219         case ACPI_TYPE_POWER:
220         case ACPI_TYPE_PROCESSOR:
221         case ACPI_TYPE_THERMAL:

223             /* These are acceptable types */
224             break;

226         case ACPI_TYPE_INTEGER:
227         case ACPI_TYPE_STRING:
228         case ACPI_TYPE_BUFFER:

230             /*
231              * These types we will allow, but we will change the type.
232              * This enables some existing code of the form:
233              *
234              *  Name (DEB, 0)
235              *  Scope (DEB) { ... }
236              */
237             ACPI_WARNING ((AE_INFO,
238                 "Type override - [%4.4s] had invalid type (%s) "
239                 "for Scope operator, changed to type ANY",
239                 "for Scope operator, changed to type ANY\n",
240                 AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type)));

242             Node->Type = ACPI_TYPE_ANY;
243             WalkState->ScopeInfo->Common.Value = ACPI_TYPE_ANY;
244             break;

246         case ACPI_TYPE_METHOD:

248             /*
249              * Allow scope change to root during execution of module-level
250              * code. Root is typed METHOD during this time.
251              */
252             if ((Node == AcpiGbl_RootNode) &&
253                 (WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
254             {
255                 break;
256             }
```

```
258             /*lint -fallthrough */

260         default:

262             /* All other types are an error */

264             ACPI_ERROR ((AE_INFO,
265                 "Invalid type (%s) for target of "
266                 "Scope operator [%4.4s] (Cannot override)",
267                 AcpiUtGetTypeName (Node->Type), AcpiUtGetNodeName (Node)));

269             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
255             return (AE_AML_OPERAND_TYPE);
270         }
271         break;

273     default:

275         /* All other opcodes */

277         if (Op && Op->Common.Node)
278         {
279             /* This op/node was previously entered into the namespace */

281             Node = Op->Common.Node;

283             if (AcpiNsOpensScope (ObjectType))
284             {
285                 Status = AcpiDsScopeStackPush (Node, ObjectType, WalkState);
286                 if (ACPI_FAILURE (Status))
287                 {
288                     return_ACPI_STATUS (Status);
289                 }
290             }

292             return_ACPI_STATUS (AE_OK);
293         }

295         /*
296          * Enter the named type into the internal namespace. We enter the name
297          * as we go downward in the parse tree. Any necessary subobjects that
298          * involve arguments to the opcode must be created as we go back up the
299          * parse tree later.
300          *
301          * Note: Name may already exist if we are executing a deferred opcode.
302          */
303         if (WalkState->DeferredNode)
304         {
305             /* This name is already in the namespace, get the node */

307             Node = WalkState->DeferredNode;
308             Status = AE_OK;
309             break;
310         }

312         Flags = ACPI_NS_NO_UPSEARCH;
313         if (WalkState->PassNumber == ACPI_IMODE_EXECUTE)
314         {
315             /* Execution mode, node cannot already exist, node is temporary */

317             Flags |= ACPI_NS_ERROR_IF_FOUND;

319             if (!(WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
320             {
321                 Flags |= ACPI_NS_TEMPORARY;
```



```
322             }
323         }

325         /* Add new entry or lookup existing entry */

327         Status = AcpiNsLookup (WalkState->ScopeInfo, BufferPtr, ObjectType,
328                     ACPI_IMODE_LOAD_PASS2, Flags, WalkState, &Node);

330         if (ACPI_SUCCESS (Status) && (Flags & ACPI_NS_TEMPORARY))
331         {
332             ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
333                 "***New Node [%4.4s] %p is temporary\n",
334                 AcpiUtGetNodeName (Node), Node));
335         }
336         break;
337     }

339     if (ACPI_FAILURE (Status))
340     {
341         ACPI_ERROR_NAMESPACE (BufferPtr, Status);
342         return_ACPI_STATUS (Status);
343     }

345     if (!Op)
346     {
347         /* Create a new op */

349         Op = AcpiPsAllocOp (WalkState->Opcode);
350         if (!Op)
351         {
352             return_ACPI_STATUS (AE_NO_MEMORY);
353         }

355         /* Initialize the new op */

357         if (Node)
358         {
359             Op->Named.Name = Node->Name.Integer;
360         }
361         *OutOp = Op;
362     }

364     /*
365      * Put the Node in the "op" object that the parser uses, so we
366      * can get it again quickly when this scope is closed
367      */
368     Op->Common.Node = Node;
369     return_ACPI_STATUS (Status);
370 }


373 /*****************************************************************************
374  *
375  * FUNCTION:    AcpiDsLoad2EndOp
376  *
377  * PARAMETERS:  WalkState       - Current state of the parse tree walk
378  *
379  * RETURN:      Status
380  *
381  * DESCRIPTION: Ascending callback used during the loading of the namespace,
382  *              both control methods and everything else.
383  *
384  ****************************************************************************/

386 ACPI_STATUS
387 AcpiDsLoad2EndOp (
```

```
388     ACPI_WALK_STATE         *WalkState)
389 {
390     ACPI_PARSE_OBJECT       *Op;
391     ACPI_STATUS             Status = AE_OK;
392     ACPI_OBJECT_TYPE        ObjectType;
393     ACPI_NAMESPACE_NODE     *Node;
394     ACPI_PARSE_OBJECT       *Arg;
395     ACPI_NAMESPACE_NODE     *NewNode;
396 #ifndef ACPI_NO_METHOD_EXECUTION
397     UINT32                  i;
398     UINT8                   RegionSpace;
399 #endif


402     ACPI_FUNCTION_TRACE (DsLoad2EndOp);

404     Op = WalkState->Op;
405     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Opcode [%s] Op %p State %p\n",
406             WalkState->OpInfo->Name, Op, WalkState));

408     /* Check if opcode had an associated namespace object */

410     if (!(WalkState->OpInfo->Flags & AML_NSOBJECT))
411     {
412         return_ACPI_STATUS (AE_OK);
413     }

415     if (Op->Common.AmlOpcode == AML_SCOPE_OP)
416     {
417         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
418             "Ending scope Op=%p State=%p\n", Op, WalkState));
419     }

421     ObjectType = WalkState->OpInfo->ObjectType;

423     /*
424      * Get the Node/name from the earlier lookup
425      * (It was saved in the *op structure)
426      */
427     Node = Op->Common.Node;

429     /*
430      * Put the Node on the object stack (Contains the ACPI Name of
431      * this object)
432      */
433     WalkState->Operands[0] = (void *) Node;
434     WalkState->NumOperands = 1;

436     /* Pop the scope stack */

438     if (AcpiNsOpensScope (ObjectType) &&
439        (Op->Common.AmlOpcode != AML_INT_METHODCALL_OP))
440     {
441         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "(%s) Popping scope for Op %p\n",
442             AcpiUtGetTypeName (ObjectType), Op));

444         Status = AcpiDsScopeStackPop (WalkState);
445         if (ACPI_FAILURE (Status))
446         {
447             goto Cleanup;
448         }
449     }

451     /*
452      * Named operations are as follows:
453      *
```



```
454      * AML_ALIAS
455      * AML_BANKFIELD
456      * AML_CREATEBITFIELD
457      * AML_CREATEBYTEFIELD
458      * AML_CREATEDWORDFIELD
459      * AML_CREATEFIELD
460      * AML_CREATEQWORDFIELD
461      * AML_CREATEWORDFIELD
462      * AML_DATA_REGION
463      * AML_DEVICE
464      * AML_EVENT
465      * AML_FIELD
466      * AML_INDEXFIELD
467      * AML_METHOD
468      * AML_METHODCALL
469      * AML_MUTEX
470      * AML_NAME
471      * AML_NAMEDFIELD
472      * AML_OPREGION
473      * AML_POWERRES
474      * AML_PROCESSOR
475      * AML_SCOPE
476      * AML_THERMALZONE
477      */

479     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
480         "Create-Load [%s] State=%p Op=%p NamedObj=%p\n",
481         AcpiPsGetOpcodeName (Op->Common.AmlOpcode), WalkState, Op, Node));

483     /* Decode the opcode */

485     Arg = Op->Common.Value.Arg;

487     switch (WalkState->OpInfo->Type)
488     {
489 #ifndef ACPI_NO_METHOD_EXECUTION

491     case AML_TYPE_CREATE_FIELD:
492         /*
493          * Create the field object, but the field buffer and index must
494          * be evaluated later during the execution phase
495          */
496         Status = AcpiDsCreateBufferField (Op, WalkState);
497         break;


499      case AML_TYPE_NAMED_FIELD:
500         /*
501          * If we are executing a method, initialize the field
502          */
503         if (WalkState->MethodNode)
504         {
505             Status = AcpiDsInitFieldObjects (Op, WalkState);
506         }

508         switch (Op->Common.AmlOpcode)
509         {
510         case AML_INDEX_FIELD_OP:

512             Status = AcpiDsCreateIndexField (Op, (ACPI_HANDLE) Arg->Common.Node,
513                         WalkState);
514             break;

516         case AML_BANK_FIELD_OP:

518             Status = AcpiDsCreateBankField (Op, Arg->Common.Node, WalkState);
```

```
519             break;
521         case AML_FIELD_OP:
523             Status = AcpiDsCreateField (Op, Arg->Common.Node, WalkState);
524             break;
526         default:
528             /* All NAMED_FIELD opcodes must be handled above */
529             break;
530         }
531         break;
533     case AML_TYPE_NAMED_SIMPLE:
535         Status = AcpiDsCreateOperands (WalkState, Arg);
536         if (ACPI_FAILURE (Status))
537         {
538             goto Cleanup;
539         }
541         switch (Op->Common.AmlOpcode)
542         {
543         case AML_PROCESSOR_OP:
545             Status = AcpiExCreateProcessor (WalkState);
546             break;
548         case AML_POWER_RES_OP:
550             Status = AcpiExCreatePowerResource (WalkState);
551             break;
553         case AML_MUTEX_OP:
555             Status = AcpiExCreateMutex (WalkState);
556             break;
558         case AML_EVENT_OP:
560             Status = AcpiExCreateEvent (WalkState);
561             break;
563         case AML_ALIAS_OP:
565             Status = AcpiExCreateAlias (WalkState);
566             break;
568         default:
570             /* Unknown opcode */
572             Status = AE_OK;
573             goto Cleanup;
574         }
576         /* Delete operands */
578         for (i = 1; i < WalkState->NumOperands; i++)
579         {
580             AcpiUtRemoveReference (WalkState->Operands[i]);
581             WalkState->Operands[i] = NULL;
582         }
```



```
584         break;
585 #endif /* ACPI_NO_METHOD_EXECUTION */
587     case AML_TYPE_NAMED_COMPLEX:
589         switch (Op->Common.AmlOpcode)
590         {
591 #ifndef ACPI_NO_METHOD_EXECUTION
592         case AML_REGION_OP:
593         case AML_DATA_REGION_OP:
595             if (Op->Common.AmlOpcode == AML_REGION_OP)
596             {
597                 RegionSpace = (ACPI_ADR_SPACE_TYPE)
598                       ((Op->Common.Value.Arg)->Common.Value.Integer);
599             }
600             else
601             {
602                 RegionSpace = ACPI_ADR_SPACE_DATA_TABLE;
603             }
605             /*
606              * The OpRegion is not fully parsed at this time. The only valid
607              * argument is the SpaceId. (We must save the address of the
608              * AML of the address and length operands)
609              *
610              * If we have a valid region, initialize it. The namespace is
611              * unlocked at this point.
612              *
613              * Need to unlock interpreter if it is locked (if we are running
614              * a control method), in order to allow _REG methods to be run
615              * during AcpiEvInitializeRegion.
616              */
617             if (WalkState->MethodNode)
618             {
619                 /*
620                  * Executing a method: initialize the region and unlock
621                  * the interpreter
622                  */
623                 Status = AcpiExCreateRegion (Op->Named.Data, Op->Named.Length,
624                             RegionSpace, WalkState);
625                 if (ACPI_FAILURE (Status))
626                 {
627                     return_ACPI_STATUS (Status);
614                     return (Status);
628                 }
630                 AcpiExExitInterpreter ();
631             }
633             Status = AcpiEvInitializeRegion (AcpiNsGetAttachedObject (Node),
634                         FALSE);
635             if (WalkState->MethodNode)
636             {
637                 AcpiExEnterInterpreter ();
638             }
640             if (ACPI_FAILURE (Status))
641             {
642                 /*
643                  *  If AE_NOT_EXIST is returned, it is not fatal
644                  *  because many regions get created before a handler
645                  *  is installed for said region.
646                  */
647                 if (AE_NOT_EXIST == Status)
```

```
                {
                    Status = AE_OK;
                }
            }
            break;


        case AML_NAME_OP:

            Status = AcpiDsCreateNode (WalkState, Node, Op);
            break;


        case AML_METHOD_OP:
            /*
             * MethodOp PkgLength NameString MethodFlags TermList
             *
             * Note: We must create the method node/object pair as soon as we
             * see the method declaration. This allows later pass1 parsing
             * of invocations of the method (need to know the number of
             * arguments.)
             */
            ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
                "LOADING-Method: State=%p Op=%p NamedObj=%p\n",
                WalkState, Op, Op->Named.Node));

            if (!AcpiNsGetAttachedObject (Op->Named.Node))
            {
                WalkState->Operands[0] = ACPI_CAST_PTR (void, Op->Named.Node);
                WalkState->NumOperands = 1;

                Status = AcpiDsCreateOperands (WalkState, Op->Common.Value.Arg);
                if (ACPI_SUCCESS (Status))
                {
                    Status = AcpiExCreateMethod (Op->Named.Data,
                                    Op->Named.Length, WalkState);
                }
                WalkState->Operands[0] = NULL;
                WalkState->NumOperands = 0;

                if (ACPI_FAILURE (Status))
                {
                    return_ACPI_STATUS (Status);
                }
            }
            break;

#endif /* ACPI_NO_METHOD_EXECUTION */

        default:

            /* All NAMED_COMPLEX opcodes must be handled above */
            break;
        }
        break;


    case AML_CLASS_INTERNAL:

        /* case AML_INT_NAMEPATH_OP: */
        break;


    case AML_CLASS_METHOD_CALL:

        ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
```



```
            "RESOLVING-MethodCall: State=%p Op=%p NamedObj=%p\n",
            WalkState, Op, Node));

        /*
         * Lookup the method name and save the Node
         */
        Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
                        ACPI_TYPE_ANY, ACPI_IMODE_LOAD_PASS2,
                        ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPEN_SCOPE,
                        WalkState, &(NewNode));
        if (ACPI_SUCCESS (Status))
        {
            /*
             * Make sure that what we found is indeed a method
             * We didn't search for a method on purpose, to see if the name
             * would resolve
             */
            if (NewNode->Type != ACPI_TYPE_METHOD)
            {
                Status = AE_AML_OPERAND_TYPE;
            }

            /* We could put the returned object (Node) on the object stack for
             * later, but for now, we will put it in the "op" object that the
             * parser uses, so we can get it again at the end of this scope
             */
            Op->Common.Node = NewNode;
        }
        else
        {
            ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
        }
        break;


    default:

        break;
    }

Cleanup:

    /* Remove the Node pushed at the very beginning */

    WalkState->Operands[0] = NULL;
    WalkState->NumOperands = 0;
    return_ACPI_STATUS (Status);
}
```

****
6978 Thu Dec 26 13:48:54 2013
new/usr/src/common/acpica/components/dispatcher/dswscope.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: dswscope - Scope stack manipulation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __DSWSCOPE_C__

#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"


#define _COMPONENT          ACPI_DISPATCHER
        ACPI_MODULE_NAME    ("dswscope")


/*****************************************************************************
 *
 * FUNCTION:    AcpiDsScopeStackClear
 *
 * PARAMETERS:  WalkState       - Current state
```



```
 *
 * RETURN:      None
 *
 * DESCRIPTION: Pop (and free) everything on the scope stack except the
 *              root scope object (which remains at the stack top.)
 *
 *****************************************************************************/

void
AcpiDsScopeStackClear (
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_GENERIC_STATE      *ScopeInfo;

    ACPI_FUNCTION_NAME (DsScopeStackClear);


    while (WalkState->ScopeInfo)
    {
        /* Pop a scope off the stack */

        ScopeInfo = WalkState->ScopeInfo;
        WalkState->ScopeInfo = ScopeInfo->Scope.Next;

        ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
            "Popped object type (%s)\n",
            AcpiUtGetTypeName (ScopeInfo->Common.Value)));
        AcpiUtDeleteGenericState (ScopeInfo);
    }
}
```

_____unchanged_portion_omitted_

****
22835 Thu Dec 26 13:48:55 2013
new/usr/src/common/acpica/components/dispatcher/dswstate.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: dswstate - Dispatcher parse tree walk management routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __DSWSTATE_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acparser.h"
  50 #include "acdispat.h"
  51 #include "acnamesp.h"

  53 #define _COMPONENT          ACPI_DISPATCHER
  54         ACPI_MODULE_NAME    ("dswstate")

  56 /* Local prototypes */

  58 static ACPI_STATUS
  59 AcpiDsResultStackPush (
```



```
  60     ACPI_WALK_STATE         *WalkState);

  62 static ACPI_STATUS
  63 AcpiDsResultStackPop (
  64     ACPI_WALK_STATE         *WalkState);


  67 /******************************************************************************
  68  *
  69  * FUNCTION:    AcpiDsResultPop
  70  *
  71  * PARAMETERS:  Object              - Where to return the popped object
  72  *              WalkState           - Current Walk state
  73  *
  74  * RETURN:      Status
  75  *
  76  * DESCRIPTION: Pop an object off the top of this walk's result stack
  77  *
  78  *****************************************************************************/

  80 ACPI_STATUS
  81 AcpiDsResultPop (
  82     ACPI_OPERAND_OBJECT     **Object,
  83     ACPI_WALK_STATE         *WalkState)
  84 {
  85     UINT32                  Index;
  86     ACPI_GENERIC_STATE      *State;
  87     ACPI_STATUS             Status;


  90     ACPI_FUNCTION_NAME (DsResultPop);


  93     State = WalkState->Results;

  95     /* Incorrect state of result stack */

  97     if (State && !WalkState->ResultCount)
  98     {
  99         ACPI_ERROR ((AE_INFO, "No results on result stack"));
 100         return (AE_AML_INTERNAL);
 101     }

 103     if (!State && WalkState->ResultCount)
 104     {
 105         ACPI_ERROR ((AE_INFO, "No result state for result stack"));
 106         return (AE_AML_INTERNAL);
 107     }

 109     /* Empty result stack */

 111     if (!State)
 112     {
 113         ACPI_ERROR ((AE_INFO, "Result stack is empty! State=%p", WalkState));
 114         return (AE_AML_NO_RETURN_VALUE);
 115     }

 117     /* Return object of the top element and clean that top element result stack

 119     WalkState->ResultCount--;
 120     Index = (UINT32) WalkState->ResultCount % ACPI_RESULTS_FRAME_OBJ_NUM;

 122     *Object = State->Results.ObjDesc [Index];
 123     if (!*Object)
 124     {
 125         ACPI_ERROR ((AE_INFO, "No result objects on result stack, State=%p",
```

```
126             WalkState));
127         return (AE_AML_NO_RETURN_VALUE);
128     }

130     State->Results.ObjDesc [Index] = NULL;
131     if (Index == 0)
132     {
133         Status = AcpiDsResultStackPop (WalkState);
134         if (ACPI_FAILURE (Status))
135         {
136             return (Status);
137         }
138     }

140     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
141         "Obj=%p [%s] Index=%X State=%p Num=%X\n", *Object,
142         AcpiUtGetObjectTypeName (*Object),
143         Index, WalkState, WalkState->ResultCount));

145     return (AE_OK);
146 }
```

unchanged_portion_omitted_

```
769 /******************************************************************************
770  *
771  * FUNCTION:    AcpiDsDeleteWalkState
772  *
773  * PARAMETERS:  WalkState       - State to delete
774  *
775  * RETURN:      Status
776  *
777  * DESCRIPTION: Delete a walk state including all internal data structures
778  *
779  *****************************************************************************/

781 void
782 AcpiDsDeleteWalkState (
783     ACPI_WALK_STATE         *WalkState)
784 {
785     ACPI_GENERIC_STATE      *State;


788     ACPI_FUNCTION_TRACE_PTR (DsDeleteWalkState, WalkState);


791     if (!WalkState)
792     {
793         return_VOID;
793         return;
794     }

796     if (WalkState->DescriptorType != ACPI_DESC_TYPE_WALK)
797     {
798         ACPI_ERROR ((AE_INFO, "%p is not a valid walk state",
799             WalkState));
800         return_VOID;
800         return;
801     }

803     /* There should not be any open scopes */

805     if (WalkState->ParserState.Scope)
806     {
807         ACPI_ERROR ((AE_INFO, "%p walk still has a scope list",
808             WalkState));
```



```
809         AcpiPsCleanupScope (&WalkState->ParserState);
810     }

812     /* Always must free any linked control states */

814     while (WalkState->ControlState)
815     {
816         State = WalkState->ControlState;
817         WalkState->ControlState = State->Common.Next;

819         AcpiUtDeleteGenericState (State);
820     }

822     /* Always must free any linked parse states */

824     while (WalkState->ScopeInfo)
825     {
826         State = WalkState->ScopeInfo;
827         WalkState->ScopeInfo = State->Common.Next;

829         AcpiUtDeleteGenericState (State);
830     }

832     /* Always must free any stacked result states */

834     while (WalkState->Results)
835     {
836         State = WalkState->Results;
837         WalkState->Results = State->Common.Next;

839         AcpiUtDeleteGenericState (State);
840     }

842     ACPI_FREE (WalkState);
843     return_VOID;
844 }
```

```
**********************************************************
    9694 Thu Dec 26 13:48:55 2013
new/usr/src/common/acpica/components/events/evevent.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evevent - Fixed Event handling and dispatch
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evevent")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/* Local prototypes */

static ACPI_STATUS
AcpiEvFixedEventInitialize (
    void);

static UINT32
```



```
AcpiEvFixedEventDispatch (
    UINT32                  Event);


/******************************************************************************
 *
 * FUNCTION:    AcpiEvInitializeEvents
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize global data structures for ACPI events (Fixed, GPE)
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvInitializeEvents (
    void)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvInitializeEvents);


    /* If Hardware Reduced flag is set, there are no fixed events */

    if (AcpiGbl_ReducedHardware)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /*
     * Initialize the Fixed and General Purpose Events. This is done prior to
     * enabling SCIs to prevent interrupts from occurring before the handlers
     * are installed.
     */
    Status = AcpiEvFixedEventInitialize ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status,
            "Unable to initialize fixed events"));
        return_ACPI_STATUS (Status);
    }

    Status = AcpiEvGpeInitialize ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status,
            "Unable to initialize general purpose events"));
        return_ACPI_STATUS (Status);
    }

    return_ACPI_STATUS (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvInstallXruptHandlers
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
```

```
 126  * DESCRIPTION: Install interrupt handlers for the SCI and Global Lock
 127  *
 128  ******************************************************************************/

 130 ACPI_STATUS
 131 AcpiEvInstallXruptHandlers (
 132     void)
 133 {
 134     ACPI_STATUS             Status;


 137     ACPI_FUNCTION_TRACE (EvInstallXruptHandlers);


 140     /* If Hardware Reduced flag is set, there is no ACPI h/w */

 142     if (AcpiGbl_ReducedHardware)
 143     {
 144         return_ACPI_STATUS (AE_OK);
 145     }

 147     /* Install the SCI handler */

 149     Status = AcpiEvInstallSciHandler ();
 150     if (ACPI_FAILURE (Status))
 151     {
 152         ACPI_EXCEPTION ((AE_INFO, Status,
 153             "Unable to install System Control Interrupt handler"));
 154         return_ACPI_STATUS (Status);
 155     }

 157     /* Install the handler for the Global Lock */

 159     Status = AcpiEvInitGlobalLockHandler ();
 160     if (ACPI_FAILURE (Status))
 161     {
 162         ACPI_EXCEPTION ((AE_INFO, Status,
 163             "Unable to initialize Global Lock handler"));
 164         return_ACPI_STATUS (Status);
 165     }

 167     AcpiGbl_EventsInitialized = TRUE;
 168     return_ACPI_STATUS (Status);
 169 }
```

*unchanged_portion_omitted_*

```
 284 /******************************************************************************
 285  *
 286  * FUNCTION:    AcpiEvFixedEventDispatch
 287  *
 288  * PARAMETERS:  Event               - Event type
 289  *
 290  * RETURN:      INTERRUPT_HANDLED or INTERRUPT_NOT_HANDLED
 291  *
 292  * DESCRIPTION: Clears the status bit for the requested event, calls the
 293  *              handler that previously registered for the event.
 294  *              NOTE: If there is no handler for the event, the event is
 295  *              disabled to prevent further interrupts.
 296  *
 297  ******************************************************************************/

 299 static UINT32
 300 AcpiEvFixedEventDispatch (
 301     UINT32                  Event)
 302 {
```



```
 304     ACPI_FUNCTION_ENTRY ();


 307     /* Clear the status bit */

 309     (void) AcpiWriteBitRegister (
 310             AcpiGbl_FixedEventInfo[Event].StatusRegisterId,
 311             ACPI_CLEAR_STATUS);

 313     /*
 314      * Make sure that a handler exists. If not, report an error
 315      * and disable the event to prevent further interrupts.
 296      * Make sure we've got a handler. If not, report an error. The event is
 297      * disabled to prevent further interrupts.
 316      */
 317     if (!AcpiGbl_FixedEventHandlers[Event].Handler)
 299     if (NULL == AcpiGbl_FixedEventHandlers[Event].Handler)
 318     {
 319         (void) AcpiWriteBitRegister (
 320                 AcpiGbl_FixedEventInfo[Event].EnableRegisterId,
 321                 ACPI_DISABLE_EVENT);

 323         ACPI_ERROR ((AE_INFO,
 324             "No installed handler for fixed event - %s (%u), disabling",
 325             AcpiUtGetEventName (Event), Event));
 306             "No installed handler for fixed event [0x%08X]",
 307             Event));

 327         return (ACPI_INTERRUPT_NOT_HANDLED);
 328     }

 330     /* Invoke the Fixed Event handler */

 332     return ((AcpiGbl_FixedEventHandlers[Event].Handler)(
 333                 AcpiGbl_FixedEventHandlers[Event].Context));
 334 }

 336 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   11425 Thu Dec 26 13:48:56 2013
new/usr/src/common/acpica/components/events/evglock.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evglock - Global Lock support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evglock")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/* Local prototypes */

static UINT32
AcpiEvGlobalLockHandler (
    void                    *Context);
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiEvInitGlobalLockHandler
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Install a handler for the global lock release event
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvInitGlobalLockHandler (
    void)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvInitGlobalLockHandler);


    /* If Hardware Reduced flag is set, there is no global lock */

    if (AcpiGbl_ReducedHardware)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /* Attempt installation of the global lock handler */

    Status = AcpiInstallFixedEventHandler (ACPI_EVENT_GLOBAL,
                AcpiEvGlobalLockHandler, NULL);

    /*
     * If the global lock does not exist on this platform, the attempt to
     * enable GBL_STATUS will fail (the GBL_ENABLE bit will not stick).
     * Map to AE_OK, but mark global lock as not present. Any attempt to
     * actually use the global lock will be flagged with an error.
     */
    AcpiGbl_GlobalLockPresent = FALSE;
    if (Status == AE_NO_HARDWARE_RESPONSE)
    {
        ACPI_ERROR ((AE_INFO,
            "No response from Global Lock hardware, disabling lock"));

        return_ACPI_STATUS (AE_OK);
    }

    Status = AcpiOsCreateLock (&AcpiGbl_GlobalLockPendingLock);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    AcpiGbl_GlobalLockPending = FALSE;
    AcpiGbl_GlobalLockPresent = TRUE;
    return_ACPI_STATUS (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvRemoveGlobalLockHandler
 *
```

```
 126  * PARAMETERS:  None
 127  *
 128  * RETURN:      Status
 129  *
 130  * DESCRIPTION: Remove the handler for the Global Lock
 131  *
 132  *****************************************************************************/

 134 ACPI_STATUS
 135 AcpiEvRemoveGlobalLockHandler (
 136     void)
 137 {
 138     ACPI_STATUS             Status;


 141     ACPI_FUNCTION_TRACE (EvRemoveGlobalLockHandler);

 143     AcpiGbl_GlobalLockPresent = FALSE;
 144     Status = AcpiRemoveFixedEventHandler (ACPI_EVENT_GLOBAL,
 145                 AcpiEvGlobalLockHandler);

 147     AcpiOsDeleteLock (AcpiGbl_GlobalLockPendingLock);
 148     return_ACPI_STATUS (Status);
 149 }
_____unchanged_portion_omitted_

 378 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   25182 Thu Dec 26 13:48:56 2013
new/usr/src/common/acpica/components/events/evgpe.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: evgpe - General Purpose Event handling and dispatch
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #include "acpi.h"
  45 #include "accommon.h"
  46 #include "acevents.h"
  47 #include "acnamesp.h"

  49 #define _COMPONENT          ACPI_EVENTS
  50         ACPI_MODULE_NAME    ("evgpe")

  52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */

  54 /* Local prototypes */

  56 static void ACPI_SYSTEM_XFACE
  57 AcpiEvAsynchExecuteGpeMethod (
  58     void                    *Context);
```



```
  60 static void ACPI_SYSTEM_XFACE
  61 AcpiEvAsynchEnableGpe (
  62     void                    *Context);


  65 /******************************************************************************
  66  *
  67  * FUNCTION:    AcpiEvUpdateGpeEnableMask
  68  *
  69  * PARAMETERS:  GpeEventInfo            - GPE to update
  70  *
  71  * RETURN:      Status
  72  *
  73  * DESCRIPTION: Updates GPE register enable mask based upon whether there are
  74  *              runtime references to this GPE
  75  *
  76  *****************************************************************************/

  78 ACPI_STATUS
  79 AcpiEvUpdateGpeEnableMask (
  80     ACPI_GPE_EVENT_INFO     *GpeEventInfo)
  81 {
  82     ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
  83     UINT32                  RegisterBit;


  86     ACPI_FUNCTION_TRACE (EvUpdateGpeEnableMask);


  89     GpeRegisterInfo = GpeEventInfo->RegisterInfo;
  90     if (!GpeRegisterInfo)
  91     {
  92         return_ACPI_STATUS (AE_NOT_EXIST);
  93     }

  95     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo);
  93     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);

  97     /* Clear the run bit up front */

  99     ACPI_CLEAR_BIT (GpeRegisterInfo->EnableForRun, RegisterBit);

 101     /* Set the mask bit only if there are references to this GPE */

 103     if (GpeEventInfo->RuntimeCount)
 104     {
 105         ACPI_SET_BIT (GpeRegisterInfo->EnableForRun, (UINT8) RegisterBit);
 106     }

 108     return_ACPI_STATUS (AE_OK);
 109 }
______unchanged_portion_omitted_


 366 /******************************************************************************
 367  *
 368  * FUNCTION:    AcpiEvGpeDetect
 369  *
 370  * PARAMETERS:  GpeXruptList        - Interrupt block for this interrupt.
 371  *                                    Can have multiple GPE blocks attached.
 372  *
 373  * RETURN:      INTERRUPT_HANDLED or INTERRUPT_NOT_HANDLED
 374  *
 375  * DESCRIPTION: Detect if any GP events have occurred. This function is
 376  *              executed at interrupt level.
```

```
 *
 ******************************************************************************/

UINT32
AcpiEvGpeDetect (
    ACPI_GPE_XRUPT_INFO     *GpeXruptList)
{
    ACPI_STATUS             Status;
    ACPI_GPE_BLOCK_INFO     *GpeBlock;
    ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
    UINT32                  IntStatus = ACPI_INTERRUPT_NOT_HANDLED;
    UINT8                   EnabledStatusByte;
    UINT32                  StatusReg;
    UINT32                  EnableReg;
    ACPI_CPU_FLAGS          Flags;
    UINT32                  i;
    UINT32                  j;


    ACPI_FUNCTION_NAME (EvGpeDetect);

    /* Check for the case where there are no GPEs */

    if (!GpeXruptList)
    {
        return (IntStatus);
    }

    /*
     * We need to obtain the GPE lock for both the data structs and registers
     * Note: Not necessary to obtain the hardware lock, since the GPE
     * registers are owned by the GpeLock.
     */
    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

    /* Examine all GPE blocks attached to this interrupt level */

    GpeBlock = GpeXruptList->GpeBlockListHead;
    while (GpeBlock)
    {
        /*
         * Read all of the 8-bit GPE status and enable registers in this GPE
         * block, saving all of them. Find all currently active GP events.
         */
        for (i = 0; i < GpeBlock->RegisterCount; i++)
        {
            /* Get the next status/enable pair */

            GpeRegisterInfo = &GpeBlock->RegisterInfo[i];

            /*
             * Optimization: If there are no GPEs enabled within this
             * register, we can safely ignore the entire register.
             */
            if (!(GpeRegisterInfo->EnableForRun |
                  GpeRegisterInfo->EnableForWake))
            {
                ACPI_DEBUG_PRINT ((ACPI_DB_INTERRUPTS,
                    "Ignore disabled registers for GPE%02X-GPE%02X: "
                    "RunEnable=%02X, WakeEnable=%02X\n",
                    GpeRegisterInfo->BaseGpeNumber,
                    GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH -
                    GpeRegisterInfo->EnableForRun,
                    GpeRegisterInfo->EnableForWake));
                continue;
            }
```



```
            /* Read the Status Register */

            Status = AcpiHwRead (&StatusReg, &GpeRegisterInfo->StatusAddress);
            if (ACPI_FAILURE (Status))
            {
                goto UnlockAndExit;
            }

            /* Read the Enable Register */

            Status = AcpiHwRead (&EnableReg, &GpeRegisterInfo->EnableAddress);
            if (ACPI_FAILURE (Status))
            {
                goto UnlockAndExit;
            }

            ACPI_DEBUG_PRINT ((ACPI_DB_INTERRUPTS,
                "Read registers for GPE%02X-GPE%02X: Status=%02X, Enable=%02X, "
                "RunEnable=%02X, WakeEnable=%02X\n",
                GpeRegisterInfo->BaseGpeNumber,
                GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH - 1),
                StatusReg, EnableReg,
                GpeRegisterInfo->EnableForRun,
                GpeRegisterInfo->EnableForWake));
                "Read GPE Register at GPE%02X: Status=%02X, Enable=%02X\n",
                GpeRegisterInfo->BaseGpeNumber, StatusReg, EnableReg));

            /* Check if there is anything active at all in this register */

            EnabledStatusByte = (UINT8) (StatusReg & EnableReg);
            if (!EnabledStatusByte)
            {
                /* No active GPEs in this register, move on */

                continue;
            }

            /* Now look at the individual GPEs in this byte register */

            for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
            {
                /* Examine one GPE bit */

                if (EnabledStatusByte & (1 << j))
                {
                    /*
                     * Found an active GPE. Dispatch the event to a handler
                     * or method.
                     */
                    IntStatus |= AcpiEvGpeDispatch (GpeBlock->Node,
                        &GpeBlock->EventInfo[((ACPI_SIZE) i *
                            ACPI_GPE_REGISTER_WIDTH) + j],
                        j + GpeRegisterInfo->BaseGpeNumber);
                }
            }
        }

        GpeBlock = GpeBlock->Next;
    }

UnlockAndExit:

    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    return (IntStatus);
}
```

```
 509 /*******************************************************************************
 510  *
 511  * FUNCTION:    AcpiEvAsynchExecuteGpeMethod
 512  *
 513  * PARAMETERS:  Context (GpeEventInfo) - Info for this GPE
 514  *
 515  * RETURN:      None
 516  *
 517  * DESCRIPTION: Perform the actual execution of a GPE control method. This
 518  *              function is called from an invocation of AcpiOsExecute and
 519  *              therefore does NOT execute at interrupt level - so that
 520  *              the control method itself is not executed in the context of
 521  *              an interrupt handler.
 522  *
 523  ******************************************************************************/

 525 static void ACPI_SYSTEM_XFACE
 526 AcpiEvAsynchExecuteGpeMethod (
 527     void                    *Context)
 528 {
 529     ACPI_GPE_EVENT_INFO     *GpeEventInfo = Context;
 530     ACPI_STATUS             Status;
 531     ACPI_GPE_EVENT_INFO     *LocalGpeEventInfo;
 532     ACPI_EVALUATE_INFO      *Info;
 533     ACPI_GPE_NOTIFY_INFO    *Notify;


 536     ACPI_FUNCTION_TRACE (EvAsynchExecuteGpeMethod);


 539     /* Allocate a local GPE block */

 541     LocalGpeEventInfo = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_EVENT_INFO));
 542     if (!LocalGpeEventInfo)
 543     {
 544         ACPI_EXCEPTION ((AE_INFO, AE_NO_MEMORY,
 545             "while handling a GPE"));
 546         return_VOID;
 547     }

 549     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
 550     if (ACPI_FAILURE (Status))
 551     {
 552         ACPI_FREE (LocalGpeEventInfo);
 553         return_VOID;
 554     }

 556     /* Must revalidate the GpeNumber/GpeBlock */

 558     if (!AcpiEvValidGpeEvent (GpeEventInfo))
 559     {
 560         Status = AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
 561         ACPI_FREE (LocalGpeEventInfo);
 562         return_VOID;
 563     }

 565     /*
 566      * Take a snapshot of the GPE info for this level - we copy the info to
 567      * prevent a race condition with RemoveHandler/RemoveBlock.
 568      */
 569     ACPI_MEMCPY (LocalGpeEventInfo, GpeEventInfo,
 570         sizeof (ACPI_GPE_EVENT_INFO));

 572     Status = AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
```



```
 573     if (ACPI_FAILURE (Status))
 574     {
 575         ACPI_FREE (LocalGpeEventInfo);
 576         return_VOID;
 577     }

 579     /* Do the correct dispatch - normal method or implicit notify */

 581     switch (LocalGpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK)
 582     {
 583     case ACPI_GPE_DISPATCH_NOTIFY:

 584         /*
 585          * Implicit notify.
 586          * Dispatch a DEVICE_WAKE notify to the appropriate handler.
 587          * NOTE: the request is queued for execution after this method
 588          * completes. The notify handlers are NOT invoked synchronously
 589          * from this thread -- because handlers may in turn run other
 590          * control methods.
 591          *
 592          * June 2012: Expand implicit notify mechanism to support
 593          * notifies on multiple device objects.
 594          */
 595         Notify = LocalGpeEventInfo->Dispatch.NotifyList;
 596         while (ACPI_SUCCESS (Status) && Notify)
 597         {
 598             Status = AcpiEvQueueNotifyRequest (Notify->DeviceNode,
 575         Status = AcpiEvQueueNotifyRequest (
 576                     LocalGpeEventInfo->Dispatch.DeviceNode,
 599                         ACPI_NOTIFY_DEVICE_WAKE);

 601             Notify = Notify->Next;
 602         }
 603         break;

 605     case ACPI_GPE_DISPATCH_METHOD:

 607         /* Allocate the evaluation information block */

 609         Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
 610         if (!Info)
 611         {
 612             Status = AE_NO_MEMORY;
 613         }
 614         else
 615         {
 616             /*
 617              * Invoke the GPE Method (_Lxx, _Exx) i.e., evaluate the
 618              * _Lxx/_Exx control method that corresponds to this GPE
 619              */
 620             Info->PrefixNode = LocalGpeEventInfo->Dispatch.MethodNode;
 621             Info->Flags = ACPI_IGNORE_RETURN_VALUE;

 623             Status = AcpiNsEvaluate (Info);
 624             ACPI_FREE (Info);
 625         }

 627         if (ACPI_FAILURE (Status))
 628         {
 629             ACPI_EXCEPTION ((AE_INFO, Status,
 630                 "while evaluating GPE method [%4.4s]",
 631                 AcpiUtGetNodeName (LocalGpeEventInfo->Dispatch.MethodNode)));
 632         }

 633         break;
```

```
635     default:

637         return_VOID; /* Should never happen */
638     }

640     /* Defer enabling of GPE until all notify handlers are done */

642     Status = AcpiOsExecute (OSL_NOTIFY_HANDLER,
643                 AcpiEvAsynchEnableGpe, LocalGpeEventInfo);
644     if (ACPI_FAILURE (Status))
645     {
646         ACPI_FREE (LocalGpeEventInfo);
647     }
648     return_VOID;
649 }
```

unchanged_portion_omitted_

```
724 /******************************************************************************
725  *
726  * FUNCTION:    AcpiEvGpeDispatch
727  *
728  * PARAMETERS:  GpeDevice           - Device node. NULL for GPE0/GPE1
729  *              GpeEventInfo        - Info for this GPE
730  *              GpeNumber           - Number relative to the parent GPE block
731  *
732  * RETURN:      INTERRUPT_HANDLED or INTERRUPT_NOT_HANDLED
733  *
734  * DESCRIPTION: Dispatch a General Purpose Event to either a function (e.g. EC)
735  *              or method (e.g. _Lxx/_Exx) handler.
736  *
737  *              This function executes at interrupt level.
738  *
739  *****************************************************************************/

741 UINT32
742 AcpiEvGpeDispatch (
743     ACPI_NAMESPACE_NODE     *GpeDevice,
744     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
745     UINT32                  GpeNumber)
746 {
747     ACPI_STATUS             Status;
748     UINT32                  ReturnValue;


751     ACPI_FUNCTION_TRACE (EvGpeDispatch);


754     /* Invoke global event handler if present */

756     AcpiGpeCount++;
757     if (AcpiGbl_GlobalEventHandler)
758     {
759         AcpiGbl_GlobalEventHandler (ACPI_EVENT_TYPE_GPE, GpeDevice,
760             GpeNumber, AcpiGbl_GlobalEventHandlerContext);
761     }

763     /*
764      * If edge-triggered, clear the GPE status bit now. Note that
765      * level-triggered events are cleared after the GPE is serviced.
766      */
767     if ((GpeEventInfo->Flags & ACPI_GPE_XRUPT_TYPE_MASK) ==
768             ACPI_GPE_EDGE_TRIGGERED)
769     {
770         Status = AcpiHwClearGpe (GpeEventInfo);
771         if (ACPI_FAILURE (Status))
```



```
772         {
773             ACPI_EXCEPTION ((AE_INFO, Status,
774                 "Unable to clear GPE%02X", GpeNumber));
775             return_UINT32 (ACPI_INTERRUPT_NOT_HANDLED);
776         }
777     }

779     /*
780      * Always disable the GPE so that it does not keep firing before
781      * any asynchronous activity completes (either from the execution
782      * of a GPE method or an asynchronous GPE handler.)
783      *
784      * If there is no handler or method to run, just disable the
785      * GPE and leave it disabled permanently to prevent further such
786      * pointless events from firing.
787      */
788     Status = AcpiHwLowSetGpe (GpeEventInfo, ACPI_GPE_DISABLE);
789     if (ACPI_FAILURE (Status))
790     {
791         ACPI_EXCEPTION ((AE_INFO, Status,
792             "Unable to disable GPE%02X", GpeNumber));
793         return_UINT32 (ACPI_INTERRUPT_NOT_HANDLED);
794     }

796     /*
797      * Dispatch the GPE to either an installed handler or the control
798      * method associated with this GPE (_Lxx or _Exx). If a handler
799      * exists, we invoke it and do not attempt to run the method.
800      * If there is neither a handler nor a method, leave the GPE
801      * disabled.
802      */
803     switch (GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK)
804     {
805     case ACPI_GPE_DISPATCH_HANDLER:

807         /* Invoke the installed handler (at interrupt level) */

809         ReturnValue = GpeEventInfo->Dispatch.Handler->Address (
810             GpeDevice, GpeNumber,
811             GpeEventInfo->Dispatch.Handler->Context);

813         /* If requested, clear (if level-triggered) and reenable the GPE */

815         if (ReturnValue & ACPI_REENABLE_GPE)
816         {
817             (void) AcpiEvFinishGpe (GpeEventInfo);
818         }
819         break;

821     case ACPI_GPE_DISPATCH_METHOD:
822     case ACPI_GPE_DISPATCH_NOTIFY:

823         /*
824          * Execute the method associated with the GPE
825          * NOTE: Level-triggered GPEs are cleared after the method completes.
826          */
827         Status = AcpiOsExecute (OSL_GPE_HANDLER,
828                     AcpiEvAsynchExecuteGpeMethod, GpeEventInfo);
829         if (ACPI_FAILURE (Status))
830         {
831             ACPI_EXCEPTION ((AE_INFO, Status,
832                 "Unable to queue handler for GPE%02X - event disabled",
833                 GpeNumber));
834         }
835         break;
```

```
837     default:
838         /*
839          * No handler or method to run!
840          * 03/2010: This case should no longer be possible. We will not allow
841          * a GPE to be enabled if it has no handler or method.
842          */
843         ACPI_ERROR ((AE_INFO,
844             "No handler or method for GPE%02X, disabling event",
845             GpeNumber));
846         break;
847     }
849     return_UINT32 (ACPI_INTERRUPT_HANDLED);
850 }
852 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   16556 Thu Dec 26 13:48:57 2013
new/usr/src/common/acpica/components/events/evgpeblk.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evgpeblk - GPE block creation and initialization.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evgpeblk")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/* Local prototypes */

static ACPI_STATUS
AcpiEvInstallGpeBlock (
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    UINT32                  InterruptNumber);

static ACPI_STATUS
AcpiEvCreateGpeInfoBlocks (
    ACPI_GPE_BLOCK_INFO     *GpeBlock);


/******************************************************************************
 *
 * FUNCTION:    AcpiEvInstallGpeBlock
 *
 * PARAMETERS:  GpeBlock                - New GPE block
 *              InterruptNumber         - Xrupt to be associated with this
 *                                        GPE block
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Install new GPE block with mutex support
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiEvInstallGpeBlock (
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    UINT32                  InterruptNumber)
{
    ACPI_GPE_BLOCK_INFO     *NextGpeBlock;
    ACPI_GPE_XRUPT_INFO     *GpeXruptBlock;
    ACPI_STATUS             Status;
    ACPI_CPU_FLAGS          Flags;


    ACPI_FUNCTION_TRACE (EvInstallGpeBlock);


    Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    Status = AcpiEvGetGpeXruptBlock (InterruptNumber, &GpeXruptBlock);
    if (ACPI_FAILURE (Status))
    GpeXruptBlock = AcpiEvGetGpeXruptBlock (InterruptNumber);
    if (!GpeXruptBlock)
    {
        Status = AE_NO_MEMORY;
        goto UnlockAndExit;
    }

    /* Install the new block at the end of the list with lock */

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
    if (GpeXruptBlock->GpeBlockListHead)
    {
        NextGpeBlock = GpeXruptBlock->GpeBlockListHead;
        while (NextGpeBlock->Next)
        {
            NextGpeBlock = NextGpeBlock->Next;
        }

        NextGpeBlock->Next = GpeBlock;
        GpeBlock->Previous = NextGpeBlock;
    }
    else
    {
```

```
122         GpeXruptBlock->GpeBlockListHead = GpeBlock;
123     }

125     GpeBlock->XruptBlock = GpeXruptBlock;
126     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);


129 UnlockAndExit:
130     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
129     Status = AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
131     return_ACPI_STATUS (Status);
132 }
______unchanged_portion_omitted_


346 /******************************************************************************
347  *
348  * FUNCTION:    AcpiEvCreateGpeBlock
349  *
350  * PARAMETERS:  GpeDevice           - Handle to the parent GPE block
351  *              GpeBlockAddress     - Address and SpaceID
352  *              RegisterCount       - Number of GPE register pairs in the block
353  *              GpeBlockBaseNumber  - Starting GPE number for the block
354  *              InterruptNumber     - H/W interrupt for the block
355  *              ReturnGpeBlock      - Where the new block descriptor is returned
356  *
357  * RETURN:      Status
358  *
359  * DESCRIPTION: Create and Install a block of GPE registers. All GPEs within
360  *              the block are disabled at exit.
361  *              Note: Assumes namespace is locked.
362  *
363  ******************************************************************************/

365 ACPI_STATUS
366 AcpiEvCreateGpeBlock (
367     ACPI_NAMESPACE_NODE     *GpeDevice,
368     ACPI_GENERIC_ADDRESS    *GpeBlockAddress,
369     UINT32                  RegisterCount,
370     UINT8                   GpeBlockBaseNumber,
371     UINT32                  InterruptNumber,
372     ACPI_GPE_BLOCK_INFO     **ReturnGpeBlock)
373 {
374     ACPI_STATUS             Status;
375     ACPI_GPE_BLOCK_INFO     *GpeBlock;
376     ACPI_GPE_WALK_INFO      WalkInfo;


379     ACPI_FUNCTION_TRACE (EvCreateGpeBlock);


382     if (!RegisterCount)
383     {
384         return_ACPI_STATUS (AE_OK);
385     }

387     /* Allocate a new GPE block */

389     GpeBlock = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_BLOCK_INFO));
390     if (!GpeBlock)
391     {
392         return_ACPI_STATUS (AE_NO_MEMORY);
393     }

395     /* Initialize the new GPE block */
```



```
397     GpeBlock->Node = GpeDevice;
398     GpeBlock->GpeCount = (UINT16) (RegisterCount * ACPI_GPE_REGISTER_WIDTH);
399     GpeBlock->Initialized = FALSE;
400     GpeBlock->RegisterCount = RegisterCount;
401     GpeBlock->BlockBaseNumber = GpeBlockBaseNumber;

403     ACPI_MEMCPY (&GpeBlock->BlockAddress, GpeBlockAddress,
404         sizeof (ACPI_GENERIC_ADDRESS));

406     /*
407      * Create the RegisterInfo and EventInfo sub-structures
408      * Note: disables and clears all GPEs in the block
409      */
410     Status = AcpiEvCreateGpeInfoBlocks (GpeBlock);
411     if (ACPI_FAILURE (Status))
412     {
413         ACPI_FREE (GpeBlock);
414         return_ACPI_STATUS (Status);
415     }

417     /* Install the new block in the global lists */

419     Status = AcpiEvInstallGpeBlock (GpeBlock, InterruptNumber);
420     if (ACPI_FAILURE (Status))
421     {
422         ACPI_FREE (GpeBlock->RegisterInfo);
423         ACPI_FREE (GpeBlock->EventInfo);
424         ACPI_FREE (GpeBlock);
425         return_ACPI_STATUS (Status);
426     }

428     AcpiGbl_AllGpesInitialized = FALSE;

430     /* Find all GPE methods (_Lxx or_Exx) for this block */

432     WalkInfo.GpeBlock = GpeBlock;
433     WalkInfo.GpeDevice = GpeDevice;
434     WalkInfo.ExecuteByOwnerId = FALSE;

436     Status = AcpiNsWalkNamespace (ACPI_TYPE_METHOD, GpeDevice,
437                 ACPI_UINT32_MAX, ACPI_NS_WALK_NO_UNLOCK,
438                 AcpiEvMatchGpeMethod, NULL, &WalkInfo, NULL);

440     /* Return the new block */

442     if (ReturnGpeBlock)
443     {
444         (*ReturnGpeBlock) = GpeBlock;
445     }

447     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
448         "    Initialized GPE %02X to %02X [%4.4s] %u regs on interrupt 0x%X\n",
444     ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
445         "GPE %02X to %02X [%4.4s] %u regs on int 0x%X\n",
449         (UINT32) GpeBlock->BlockBaseNumber,
450         (UINT32) (GpeBlock->BlockBaseNumber + (GpeBlock->GpeCount - 1)),
451         GpeDevice->Name.Ascii, GpeBlock->RegisterCount,
452         InterruptNumber));

454     /* Update global count of currently available GPEs */

456     AcpiCurrentGpeCount += GpeBlock->GpeCount;
457     return_ACPI_STATUS (AE_OK);
458 }
```

```
461 /*******************************************************************************
462  *
463  * FUNCTION:    AcpiEvInitializeGpeBlock
464  *
465  * PARAMETERS:  ACPI_GPE_CALLBACK
466  *
467  * RETURN:      Status
468  *
469  * DESCRIPTION: Initialize and enable a GPE block. Enable GPEs that have
470  *              associated methods.
471  *              Note: Assumes namespace is locked.
472  *
473  ******************************************************************************/

475 ACPI_STATUS
476 AcpiEvInitializeGpeBlock (
477     ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
478     ACPI_GPE_BLOCK_INFO     *GpeBlock,
479     void                    *Ignored)
480 {
481     ACPI_STATUS             Status;
482     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
483     UINT32                  GpeEnabledCount;
484     UINT32                  GpeIndex;
485     UINT32                  i;
486     UINT32                  j;


489     ACPI_FUNCTION_TRACE (EvInitializeGpeBlock);


492     /*
493      * Ignore a null GPE block (e.g., if no GPE block 1 exists), and
494      * any GPE blocks that have been initialized already.
495      */
496     if (!GpeBlock || GpeBlock->Initialized)
497     {
498         return_ACPI_STATUS (AE_OK);
499     }

501     /*
502      * Enable all GPEs that have a corresponding method and have the
503      * ACPI_GPE_CAN_WAKE flag unset. Any other GPEs within this block
504      * must be enabled via the acpi_enable_gpe() interface.
505      */
506     GpeEnabledCount = 0;

508     for (i = 0; i < GpeBlock->RegisterCount; i++)
509     {
510         for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
511         {
512             /* Get the info block for this particular GPE */

514             GpeIndex = (i * ACPI_GPE_REGISTER_WIDTH) + j;
515             GpeEventInfo = &GpeBlock->EventInfo[GpeIndex];

517             /*
518              * Ignore GPEs that have no corresponding _Lxx/_Exx method
519              * and GPEs that are used to wake the system
520              */
521             if (((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) == ACPI_GPE_DISP
522                 ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) == ACPI_GPE_DISP
523                 (GpeEventInfo->Flags & ACPI_GPE_CAN_WAKE))
524             {
525                 continue;
526             }
```



```
528             Status = AcpiEvAddGpeReference (GpeEventInfo);
529             if (ACPI_FAILURE (Status))
530             {
531                 ACPI_EXCEPTION ((AE_INFO, Status,
532                     "Could not enable GPE 0x%02X",
533                     GpeIndex + GpeBlock->BlockBaseNumber));
534                 continue;
535             }

537             GpeEnabledCount++;
538         }
539     }

541     if (GpeEnabledCount)
542     {
543         ACPI_INFO ((AE_INFO,
544             "Enabled %u GPEs in block %02X to %02X", GpeEnabledCount,
545             (UINT32) GpeBlock->BlockBaseNumber,
546             (UINT32) (GpeBlock->BlockBaseNumber + (GpeBlock->GpeCount - 1))));
540         ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
541             "Enabled %u GPEs in this block\n", GpeEnabledCount));
547     }

549     GpeBlock->Initialized = TRUE;
550     return_ACPI_STATUS (AE_OK);
551 }

553 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   14987 Thu Dec 26 13:48:57 2013
new/usr/src/common/acpica/components/events/evgpeinit.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evgpeinit - System GPE initialization and update
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evgpeinit")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/*
 * Note: History of _PRW support in ACPICA
 *
 * Originally (2000 - 2010), the GPE initialization code performed a walk of
 * the entire namespace to execute the _PRW methods and detect all GPEs
```



```
 * capable of waking the system.
 *
 * As of 10/2010, the _PRW method execution has been removed since it is
 * actually unnecessary. The host OS must in fact execute all _PRW methods
 * in order to identify the device/power-resource dependencies. We now put
 * the onus on the host OS to identify the wake GPEs as part of this process
 * and to inform ACPICA of these GPEs via the AcpiSetupGpeForWake interface. Thi
 * not only reduces the complexity of the ACPICA initialization code, but in
 * some cases (on systems with very large namespaces) it should reduce the
 * kernel boot time as well.
 */

/******************************************************************************
 *
 * FUNCTION:    AcpiEvGpeInitialize
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize the GPE data structures and the FADT GPE 0/1 blocks
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvGpeInitialize (
    void)
{
    UINT32                  RegisterCount0 = 0;
    UINT32                  RegisterCount1 = 0;
    UINT32                  GpeNumberMax = 0;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvGpeInitialize);


    ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
        "Initializing General Purpose Events (GPEs):\n"));

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * Initialize the GPE Block(s) defined in the FADT
     *
     * Why the GPE register block lengths are divided by 2:  From the ACPI
     * Spec, section "General-Purpose Event Registers", we have:
     *
     * "Each register block contains two registers of equal length
     *  GPEx_STS and GPEx_EN (where x is 0 or 1). The length of the
     *  GPE0_STS and GPE0_EN registers is equal to half the GPE0_LEN
     *  The length of the GPE1_STS and GPE1_EN registers is equal to
     *  half the GPE1_LEN. If a generic register block is not supported
     *  then its respective block pointer and block length values in the
     *  FADT table contain zeros. The GPE0_LEN and GPE1_LEN do not need
     *  to be the same size."
     */

    /*
     * Determine the maximum GPE number for this machine.
     *
     * Note: both GPE0 and GPE1 are optional, and either can exist without
```

```
126      * the other.
127      *
128      * If EITHER the register length OR the block address are zero, then that
129      * particular block is not supported.
130      */
131     if (AcpiGbl_FADT.Gpe0BlockLength &&
132         AcpiGbl_FADT.XGpe0Block.Address)
133     {
134         /* GPE block 0 exists (has both length and address > 0) */

136         RegisterCount0 = (UINT16) (AcpiGbl_FADT.Gpe0BlockLength / 2);

137         GpeNumberMax = (RegisterCount0 * ACPI_GPE_REGISTER_WIDTH) - 1;

139         /* Install GPE Block 0 */

141         Status = AcpiEvCreateGpeBlock (AcpiGbl_FadtGpeDevice,
142                     &AcpiGbl_FADT.XGpe0Block, RegisterCount0, 0,
143                     AcpiGbl_FADT.SciInterrupt, &AcpiGbl_GpeFadtBlocks[0]);

145         if (ACPI_FAILURE (Status))
146         {
147             ACPI_EXCEPTION ((AE_INFO, Status,
148                 "Could not create GPE Block 0"));
149         }
150     }

152     if (AcpiGbl_FADT.Gpe1BlockLength &&
153         AcpiGbl_FADT.XGpe1Block.Address)
154     {
155         /* GPE block 1 exists (has both length and address > 0) */

157         RegisterCount1 = (UINT16) (AcpiGbl_FADT.Gpe1BlockLength / 2);

159         /* Check for GPE0/GPE1 overlap (if both banks exist) */

161         if ((RegisterCount0) &&
162             (GpeNumberMax >= AcpiGbl_FADT.Gpe1Base))
163         {
164             ACPI_ERROR ((AE_INFO,
165                 "GPE0 block (GPE 0 to %u) overlaps the GPE1 block "
166                 "(GPE %u to %u) - Ignoring GPE1",
167                 GpeNumberMax, AcpiGbl_FADT.Gpe1Base,
168                 AcpiGbl_FADT.Gpe1Base +
169                 ((RegisterCount1 * ACPI_GPE_REGISTER_WIDTH) - 1)));

171             /* Ignore GPE1 block by setting the register count to zero */

173             RegisterCount1 = 0;
174         }
175         else
176         {
177             /* Install GPE Block 1 */

179             Status = AcpiEvCreateGpeBlock (AcpiGbl_FadtGpeDevice,
180                         &AcpiGbl_FADT.XGpe1Block, RegisterCount1,
181                         AcpiGbl_FADT.Gpe1Base,
182                         AcpiGbl_FADT.SciInterrupt, &AcpiGbl_GpeFadtBlocks[1]);

184             if (ACPI_FAILURE (Status))
185             {
186                 ACPI_EXCEPTION ((AE_INFO, Status,
187                     "Could not create GPE Block 1"));
188             }

190             /*
```



```
191              * GPE0 and GPE1 do not have to be contiguous in the GPE number
192              * space. However, GPE0 always starts at GPE number zero.
193              */
194             GpeNumberMax = AcpiGbl_FADT.Gpe1Base +
195                            ((RegisterCount1 * ACPI_GPE_REGISTER_WIDTH) - 1);
196         }
197     }

199     /* Exit if there are no GPE registers */

201     if ((RegisterCount0 + RegisterCount1) == 0)
202     {
203         /* GPEs are not required by ACPI, this is OK */

205         ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
206             "There are no GPE blocks defined in the FADT\n"));
207         Status = AE_OK;
208         goto Cleanup;
209     }

208     /* Check for Max GPE number out-of-range */

210     if (GpeNumberMax > ACPI_GPE_MAX)
211     {
212         ACPI_ERROR ((AE_INFO,
213             "Maximum GPE number from FADT is too large: 0x%X",
214             GpeNumberMax));
215         Status = AE_BAD_VALUE;
216         goto Cleanup;
217     }

212 Cleanup:
213     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
214     return_ACPI_STATUS (AE_OK);
215 }
```

_____unchanged_portion_omitted_

```
300 /******************************************************************************
301  *
302  * FUNCTION:    AcpiEvMatchGpeMethod
303  *
304  * PARAMETERS:  Callback from WalkNamespace
305  *
306  * RETURN:      Status
307  *
308  * DESCRIPTION: Called from AcpiWalkNamespace. Expects each object to be a
309  *              control method under the _GPE portion of the namespace.
310  *              Extract the name and GPE type from the object, saving this
311  *              information for quick lookup during GPE dispatch. Allows a
312  *              per-OwnerId evaluation if ExecuteByOwnerId is TRUE in the
313  *              WalkInfo parameter block.
314  *
315  *              The name of each GPE control method is of the form:
316  *              "_Lxx" or "_Exx", where:
317  *                  L      - means that the GPE is level triggered
318  *                  E      - means that the GPE is edge triggered
319  *                  xx     - is the GPE number [in HEX]
320  *
321  * If WalkInfo->ExecuteByOwnerId is TRUE, we only execute examine GPE methods
322  * with that owner.
323  *
324  *****************************************************************************/

326 ACPI_STATUS
327 AcpiEvMatchGpeMethod (
```

```
328     ACPI_HANDLE             ObjHandle,
329     UINT32                  Level,
330     void                    *Context,
331     void                    **ReturnValue)
332 {
333     ACPI_NAMESPACE_NODE     *MethodNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Ob
334     ACPI_GPE_WALK_INFO      *WalkInfo = ACPI_CAST_PTR (ACPI_GPE_WALK_INFO, Conte
335     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
336     UINT32                  GpeNumber;
337     char                    Name[ACPI_NAME_SIZE + 1];
338     UINT8                   Type;


341     ACPI_FUNCTION_TRACE (EvMatchGpeMethod);


344     /* Check if requested OwnerId matches this OwnerId */

346     if ((WalkInfo->ExecuteByOwnerId) &&
347         (MethodNode->OwnerId != WalkInfo->OwnerId))
348     {
349         return_ACPI_STATUS (AE_OK);
350     }

352     /*
353      * Match and decode the _Lxx and _Exx GPE method names
354      *
355      * 1) Extract the method name and null terminate it
356      */
357     ACPI_MOVE_32_TO_32 (Name, &MethodNode->Name.Integer);
358     Name[ACPI_NAME_SIZE] = 0;

360     /* 2) Name must begin with an underscore */

362     if (Name[0] != '_')
363     {
364         return_ACPI_STATUS (AE_OK); /* Ignore this method */
365     }

367     /*
368      * 3) Edge/Level determination is based on the 2nd character
369      *    of the method name
370      */
371     switch (Name[1])
372     {
373     case 'L':

375         Type = ACPI_GPE_LEVEL_TRIGGERED;
376         break;

378     case 'E':

380         Type = ACPI_GPE_EDGE_TRIGGERED;
381         break;

383     default:

385         /* Unknown method type, just ignore it */

387         ACPI_DEBUG_PRINT ((ACPI_DB_LOAD,
388             "Ignoring unknown GPE method type: %s "
389             "(name not of form _Lxx or _Exx)", Name));
390         return_ACPI_STATUS (AE_OK);
391     }

393     /* 4) The last two characters of the name are the hex GPE Number */
```



```
395     GpeNumber = ACPI_STRTOUL (&Name[2], NULL, 16);
396     if (GpeNumber == ACPI_UINT32_MAX)
397     {
398         /* Conversion failed; invalid method, just ignore it */

400         ACPI_DEBUG_PRINT ((ACPI_DB_LOAD,
401             "Could not extract GPE number from name: %s "
402             "(name is not of form _Lxx or _Exx)", Name));
403         return_ACPI_STATUS (AE_OK);
404     }

406     /* Ensure that we have a valid GPE number for this GPE block */

408     GpeEventInfo = AcpiEvLowGetGpeInfo (GpeNumber, WalkInfo->GpeBlock);
409     if (!GpeEventInfo)
410     {
411         /*
412          * This GpeNumber is not valid for this GPE block, just ignore it.
413          * However, it may be valid for a different GPE block, since GPE0
414          * and GPE1 methods both appear under \_GPE.
415          */
416         return_ACPI_STATUS (AE_OK);
417     }

419     if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
420             ACPI_GPE_DISPATCH_HANDLER)
421     {
422         /* If there is already a handler, ignore this GPE method */

424         return_ACPI_STATUS (AE_OK);
425     }

427     if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
428             ACPI_GPE_DISPATCH_METHOD)
429     {
430         /*
431          * If there is already a method, ignore this method. But check
432          * for a type mismatch (if both the _Lxx AND _Exx exist)
433          */
434         if (Type != (GpeEventInfo->Flags & ACPI_GPE_XRUPT_TYPE_MASK))
435         {
436             ACPI_ERROR ((AE_INFO,
437                 "For GPE 0x%.2X, found both _L%2.2X and _E%2.2X methods",
438                 GpeNumber, GpeNumber, GpeNumber));
439         }
440         return_ACPI_STATUS (AE_OK);
441     }

443     /*
444      * Add the GPE information from above to the GpeEventInfo block for
445      * use during dispatch of this GPE.
446      */
447     GpeEventInfo->Flags &= ~(ACPI_GPE_DISPATCH_MASK);
448     GpeEventInfo->Flags |= (UINT8) (Type | ACPI_GPE_DISPATCH_METHOD);
449     GpeEventInfo->Dispatch.MethodNode = MethodNode;

451     ACPI_DEBUG_PRINT ((ACPI_DB_LOAD,
452         "Registered GPE method %s as GPE number 0x%.2X\n",
453         Name, GpeNumber));
454     return_ACPI_STATUS (AE_OK);
455 }

457 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   12952 Thu Dec 26 13:48:57 2013
new/usr/src/common/acpica/components/events/evgpeutil.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evgpeutil - GPE utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acevents.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evgpeutil")


#if (!ACPI_REDUCED_HARDWARE) /* Entire module */
/******************************************************************************
 *
 * FUNCTION:    AcpiEvWalkGpeList
 *
 * PARAMETERS:  GpeWalkCallback     - Routine called for each GPE block
```



```
 *              Context             - Value passed to callback
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk the GPE lists.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEvWalkGpeList (
    ACPI_GPE_CALLBACK       GpeWalkCallback,
    void                    *Context)
{
    ACPI_GPE_BLOCK_INFO     *GpeBlock;
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo;
    ACPI_STATUS             Status = AE_OK;
    ACPI_CPU_FLAGS          Flags;


    ACPI_FUNCTION_TRACE (EvWalkGpeList);


    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

    /* Walk the interrupt level descriptor list */

    GpeXruptInfo = AcpiGbl_GpeXruptListHead;
    while (GpeXruptInfo)
    {
        /* Walk all Gpe Blocks attached to this interrupt level */

        GpeBlock = GpeXruptInfo->GpeBlockListHead;
        while (GpeBlock)
        {
            /* One callback per GPE block */

            Status = GpeWalkCallback (GpeXruptInfo, GpeBlock, Context);
            if (ACPI_FAILURE (Status))
            {
                if (Status == AE_CTRL_END) /* Callback abort */
                {
                    Status = AE_OK;
                }
                goto UnlockAndExit;
            }

            GpeBlock = GpeBlock->Next;
        }

        GpeXruptInfo = GpeXruptInfo->Next;
    }

UnlockAndExit:
    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    return_ACPI_STATUS (Status);
}
______unchanged_portion_omitted_


/******************************************************************************
 *
 * FUNCTION:    AcpiEvGetGpeXruptBlock
 *
 * PARAMETERS:  InterruptNumber             - Interrupt for a GPE block
 *              GpeXruptBlock               - Where the block is returned
 *
```

```
 * RETURN:      Status
 * RETURN:      A GPE interrupt block
 *
 * DESCRIPTION: Get or Create a GPE interrupt block. There is one interrupt
 *              block per unique interrupt level used for GPEs. Should be
 *              called only when the GPE lists are semaphore locked and not
 *              subject to change.
 *
 ******************************************************************************/

ACPI_STATUS
ACPI_GPE_XRUPT_INFO *
AcpiEvGetGpeXruptBlock (
    UINT32                  InterruptNumber,
    ACPI_GPE_XRUPT_INFO     **GpeXruptBlock)
    UINT32                  InterruptNumber)
{
    ACPI_GPE_XRUPT_INFO     *NextGpeXrupt;
    ACPI_GPE_XRUPT_INFO     *GpeXrupt;
    ACPI_STATUS             Status;
    ACPI_CPU_FLAGS          Flags;


    ACPI_FUNCTION_TRACE (EvGetGpeXruptBlock);


    /* No need for lock since we are not changing any list elements here */

    NextGpeXrupt = AcpiGbl_GpeXruptListHead;
    while (NextGpeXrupt)
    {
        if (NextGpeXrupt->InterruptNumber == InterruptNumber)
        {
            *GpeXruptBlock = NextGpeXrupt;
            return_ACPI_STATUS (AE_OK);
            return_PTR (NextGpeXrupt);
        }

        NextGpeXrupt = NextGpeXrupt->Next;
    }

    /* Not found, must allocate a new xrupt descriptor */

    GpeXrupt = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_XRUPT_INFO));
    if (!GpeXrupt)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
        return_PTR (NULL);
    }

    GpeXrupt->InterruptNumber = InterruptNumber;

    /* Install new interrupt descriptor with spin lock */

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
    if (AcpiGbl_GpeXruptListHead)
    {
        NextGpeXrupt = AcpiGbl_GpeXruptListHead;
        while (NextGpeXrupt->Next)
        {
            NextGpeXrupt = NextGpeXrupt->Next;
        }

        NextGpeXrupt->Next = GpeXrupt;
        GpeXrupt->Previous = NextGpeXrupt;
    }
```



```
    else
    {
        AcpiGbl_GpeXruptListHead = GpeXrupt;
    }

    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);

    /* Install new interrupt handler if not SCI_INT */

    if (InterruptNumber != AcpiGbl_FADT.SciInterrupt)
    {
        Status = AcpiOsInstallInterruptHandler (InterruptNumber,
                    AcpiEvGpeXruptHandler, GpeXrupt);
        if (ACPI_FAILURE (Status))
        {
            ACPI_EXCEPTION ((AE_INFO, Status,
            ACPI_ERROR ((AE_INFO,
                "Could not install GPE interrupt handler at level 0x%X",
                InterruptNumber));
            return_ACPI_STATUS (Status);
            return_PTR (NULL);
        }
    }

    *GpeXruptBlock = GpeXrupt;
    return_ACPI_STATUS (AE_OK);
    return_PTR (GpeXrupt);
}
_____unchanged_portion_omitted_


/******************************************************************************
 *
 * FUNCTION:    AcpiEvDeleteGpeHandlers
 *
 * PARAMETERS:  GpeXruptInfo        - GPE Interrupt info
 *              GpeBlock            - Gpe Block info
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Delete all Handler objects found in the GPE data structs.
 *              Used only prior to termination.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEvDeleteGpeHandlers (
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    void                    *Context)
{
    ACPI_GPE_EVENT_INFO     *GpeEventInfo;
    ACPI_GPE_NOTIFY_INFO    *Notify;
    ACPI_GPE_NOTIFY_INFO    *Next;
    UINT32                  i;
    UINT32                  j;


    ACPI_FUNCTION_TRACE (EvDeleteGpeHandlers);


    /* Examine each GPE Register within the block */

    for (i = 0; i < GpeBlock->RegisterCount; i++)
    {
        /* Now look at the individual GPEs in this byte register */
```

```
414         for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
415         {
416             GpeEventInfo = &GpeBlock->EventInfo[((ACPI_SIZE) i *
417                 ACPI_GPE_REGISTER_WIDTH) + j];

419             if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
420                     ACPI_GPE_DISPATCH_HANDLER)
421             {
422                 /* Delete an installed handler block */

424                 ACPI_FREE (GpeEventInfo->Dispatch.Handler);
425                 GpeEventInfo->Dispatch.Handler = NULL;
426                 GpeEventInfo->Flags &= ~ACPI_GPE_DISPATCH_MASK;
427             }
428             else if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
429                     ACPI_GPE_DISPATCH_NOTIFY)
430             {
431                 /* Delete the implicit notification device list */

433                 Notify = GpeEventInfo->Dispatch.NotifyList;
434                 while (Notify)
435                 {
436                     Next = Notify->Next;
437                     ACPI_FREE (Notify);
438                     Notify = Next;
439                 }
440                 GpeEventInfo->Dispatch.NotifyList = NULL;
441                 GpeEventInfo->Flags &= ~ACPI_GPE_DISPATCH_MASK;
442             }
443         }
444     }

446     return_ACPI_STATUS (AE_OK);
447 }

449 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   17268 Thu Dec 26 13:48:58 2013
new/usr/src/common/acpica/components/events/evhandler.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evhandler - Support for Address Space handlers
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EVHANDLER_C__

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evhandler")


/* Local prototypes */

static ACPI_STATUS
AcpiEvInstallHandler (
    ACPI_HANDLE             ObjHandle,
```



```
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

/* These are the address spaces that will get default handlers */

UINT8        AcpiGbl_DefaultAddressSpaces[ACPI_NUM_DEFAULT_SPACES] =
{
    ACPI_ADR_SPACE_SYSTEM_MEMORY,
    ACPI_ADR_SPACE_SYSTEM_IO,
    ACPI_ADR_SPACE_PCI_CONFIG,
    ACPI_ADR_SPACE_DATA_TABLE
};


/******************************************************************************
 *
 * FUNCTION:    AcpiEvInstallRegionHandlers
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Installs the core subsystem default address space handlers.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvInstallRegionHandlers (
    void)
{
    ACPI_STATUS             Status;
    UINT32                  i;


    ACPI_FUNCTION_TRACE (EvInstallRegionHandlers);


    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * All address spaces (PCI Config, EC, SMBus) are scope dependent and
     * registration must occur for a specific device.
     *
     * In the case of the system memory and IO address spaces there is
     * currently no device associated with the address space. For these we
     * use the root.
     *
     * We install the default PCI config space handler at the root so that
     * this space is immediately available even though the we have not
     * enumerated all the PCI Root Buses yet. This is to conform to the ACPI
     * specification which states that the PCI config space must be always
     * available -- even though we are nowhere near ready to find the PCI root
     * buses at this point.
     *
     * NOTE: We ignore AE_ALREADY_EXISTS because this means that a handler
     * has already been installed (via AcpiInstallAddressSpaceHandler).
     * Similar for AE_SAME_HANDLER.
     */
    for (i = 0; i < ACPI_NUM_DEFAULT_SPACES; i++)
    {
        Status = AcpiEvInstallSpaceHandler (AcpiGbl_RootNode,
```

```
128                     AcpiGbl_DefaultAddressSpaces[i],
129                     ACPI_DEFAULT_HANDLER, NULL, NULL);
130         switch (Status)
131         {
132         case AE_OK:
133         case AE_SAME_HANDLER:
134         case AE_ALREADY_EXISTS:

136             /* These exceptions are all OK */

138             Status = AE_OK;
139             break;

141         default:

143             goto UnlockAndExit;
144         }
145     }

147 UnlockAndExit:
148     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
149     return_ACPI_STATUS (Status);
150 }


153 /*******************************************************************************
154  *
155  * FUNCTION:    AcpiEvHasDefaultHandler
156  *
157  * PARAMETERS:  Node                - Namespace node for the device
158  *              SpaceId             - The address space ID
159  *
160  * RETURN:      TRUE if default handler is installed, FALSE otherwise
161  *
162  * DESCRIPTION: Check if the default handler is installed for the requested
163  *              space ID.
164  *
165  ******************************************************************************/

167 BOOLEAN
168 AcpiEvHasDefaultHandler (
169     ACPI_NAMESPACE_NODE     *Node,
170     ACPI_ADR_SPACE_TYPE     SpaceId)
171 {
172     ACPI_OPERAND_OBJECT     *ObjDesc;
173     ACPI_OPERAND_OBJECT     *HandlerObj;


176     /* Must have an existing internal object */

178     ObjDesc = AcpiNsGetAttachedObject (Node);
179     if (ObjDesc)
180     {
181         HandlerObj = ObjDesc->Device.Handler;

183         /* Walk the linked list of handlers for this object */

185         while (HandlerObj)
186         {
187             if (HandlerObj->AddressSpace.SpaceId == SpaceId)
188             {
189                 if (HandlerObj->AddressSpace.HandlerFlags &
190                         ACPI_ADDR_HANDLER_DEFAULT_INSTALLED)
191                 {
192                     return (TRUE);
193                 }
```



```
194             }

196             HandlerObj = HandlerObj->AddressSpace.Next;
197         }
198     }

200     return (FALSE);
201 }


204 /*******************************************************************************
205  *
206  * FUNCTION:    AcpiEvInstallHandler
207  *
208  * PARAMETERS:  WalkNamespace callback
209  *
210  * DESCRIPTION: This routine installs an address handler into objects that are
211  *              of type Region or Device.
212  *
213  *              If the Object is a Device, and the device has a handler of
214  *              the same type then the search is terminated in that branch.
215  *
216  *              This is because the existing handler is closer in proximity
217  *              to any more regions than the one we are trying to install.
218  *
219  ******************************************************************************/

221 static ACPI_STATUS
222 AcpiEvInstallHandler (
223     ACPI_HANDLE             ObjHandle,
224     UINT32                  Level,
225     void                    *Context,
226     void                    **ReturnValue)
227 {
228     ACPI_OPERAND_OBJECT     *HandlerObj;
229     ACPI_OPERAND_OBJECT     *NextHandlerObj;
230     ACPI_OPERAND_OBJECT     *ObjDesc;
231     ACPI_NAMESPACE_NODE     *Node;
232     ACPI_STATUS             Status;


235     ACPI_FUNCTION_NAME (EvInstallHandler);


238     HandlerObj = (ACPI_OPERAND_OBJECT  *) Context;

240     /* Parameter validation */

242     if (!HandlerObj)
243     {
244         return (AE_OK);
245     }

247     /* Convert and validate the device handle */

249     Node = AcpiNsValidateHandle (ObjHandle);
250     if (!Node)
251     {
252         return (AE_BAD_PARAMETER);
253     }

255     /*
256      * We only care about regions and objects that are allowed to have
257      * address space handlers
258      */
259     if ((Node->Type != ACPI_TYPE_DEVICE) &&
```

```
260         (Node->Type != ACPI_TYPE_REGION) &&
261         (Node != AcpiGbl_RootNode))
262     {
263         return (AE_OK);
264     }

266     /* Check for an existing internal object */

268     ObjDesc = AcpiNsGetAttachedObject (Node);
269     if (!ObjDesc)
270     {
271         /* No object, just exit */

273         return (AE_OK);
274     }

276     /* Devices are handled different than regions */

278     if (ObjDesc->Common.Type == ACPI_TYPE_DEVICE)
279     {
280         /* Check if this Device already has a handler for this address space */

282         NextHandlerObj = ObjDesc->Device.Handler;
283         while (NextHandlerObj)
284         {
285             /* Found a handler, is it for the same address space? */

287             if (NextHandlerObj->AddressSpace.SpaceId ==
288                     HandlerObj->AddressSpace.SpaceId)
289             {
290                 ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
291                     "Found handler for region [%s] in device %p(%p) "
292                     "handler %p\n",
293                     AcpiUtGetRegionName (HandlerObj->AddressSpace.SpaceId),
294                     ObjDesc, NextHandlerObj, HandlerObj));

296                 /*
297                  * Since the object we found it on was a device, then it
298                  * means that someone has already installed a handler for
299                  * the branch of the namespace from this device on. Just
300                  * bail out telling the walk routine to not traverse this
301                  * branch. This preserves the scoping rule for handlers.
302                  */
303                 return (AE_CTRL_DEPTH);
304             }

306             /* Walk the linked list of handlers attached to this device */

308             NextHandlerObj = NextHandlerObj->AddressSpace.Next;
309         }

311         /*
312          * As long as the device didn't have a handler for this space we
313          * don't care about it. We just ignore it and proceed.
314          */
315         return (AE_OK);
316     }

318     /* Object is a Region */

320     if (ObjDesc->Region.SpaceId != HandlerObj->AddressSpace.SpaceId)
321     {
322         /* This region is for a different address space, just ignore it */

324         return (AE_OK);
325     }
```



```
327     /*
328      * Now we have a region and it is for the handler's address space type.
329      *
330      * First disconnect region for any previous handler (if any)
331      */
332     AcpiEvDetachRegion (ObjDesc, FALSE);

334     /* Connect the region to the new handler */

336     Status = AcpiEvAttachRegion (HandlerObj, ObjDesc, FALSE);
337     return (Status);
338 }


341 /*******************************************************************************
342  *
343  * FUNCTION:    AcpiEvInstallSpaceHandler
344  *
345  * PARAMETERS:  Node            - Namespace node for the device
346  *              SpaceId         - The address space ID
347  *              Handler         - Address of the handler
348  *              Setup           - Address of the setup function
349  *              Context         - Value passed to the handler on each access
350  *
351  * RETURN:      Status
352  *
353  * DESCRIPTION: Install a handler for all OpRegions of a given SpaceId.
354  *              Assumes namespace is locked
355  *
356  ******************************************************************************/

358 ACPI_STATUS
359 AcpiEvInstallSpaceHandler (
360     ACPI_NAMESPACE_NODE     *Node,
361     ACPI_ADR_SPACE_TYPE     SpaceId,
362     ACPI_ADR_SPACE_HANDLER  Handler,
363     ACPI_ADR_SPACE_SETUP    Setup,
364     void                    *Context)
365 {
366     ACPI_OPERAND_OBJECT     *ObjDesc;
367     ACPI_OPERAND_OBJECT     *HandlerObj;
368     ACPI_STATUS             Status;
369     ACPI_OBJECT_TYPE        Type;
370     UINT8                   Flags = 0;


373     ACPI_FUNCTION_TRACE (EvInstallSpaceHandler);


376     /*
377      * This registration is valid for only the types below and the root. This
378      * is where the default handlers get placed.
379      */
380     if ((Node->Type != ACPI_TYPE_DEVICE)     &&
381         (Node->Type != ACPI_TYPE_PROCESSOR)  &&
382         (Node->Type != ACPI_TYPE_THERMAL)    &&
383         (Node != AcpiGbl_RootNode))
384     {
385         Status = AE_BAD_PARAMETER;
386         goto UnlockAndExit;
387     }

389     if (Handler == ACPI_DEFAULT_HANDLER)
390     {
391         Flags = ACPI_ADDR_HANDLER_DEFAULT_INSTALLED;
```

```
393         switch (SpaceId)
394         {
395         case ACPI_ADR_SPACE_SYSTEM_MEMORY:

397             Handler = AcpiExSystemMemorySpaceHandler;
398             Setup   = AcpiEvSystemMemoryRegionSetup;
399             break;

401         case ACPI_ADR_SPACE_SYSTEM_IO:

403             Handler = AcpiExSystemIoSpaceHandler;
404             Setup   = AcpiEvIoSpaceRegionSetup;
405             break;

407         case ACPI_ADR_SPACE_PCI_CONFIG:

409             Handler = AcpiExPciConfigSpaceHandler;
410             Setup   = AcpiEvPciConfigRegionSetup;
411             break;

413         case ACPI_ADR_SPACE_CMOS:

415             Handler = AcpiExCmosSpaceHandler;
416             Setup   = AcpiEvCmosRegionSetup;
417             break;

419         case ACPI_ADR_SPACE_PCI_BAR_TARGET:

421             Handler = AcpiExPciBarSpaceHandler;
422             Setup   = AcpiEvPciBarRegionSetup;
423             break;

425         case ACPI_ADR_SPACE_DATA_TABLE:

427             Handler = AcpiExDataTableSpaceHandler;
428             Setup   = NULL;
429             break;

431         default:

433             Status = AE_BAD_PARAMETER;
434             goto UnlockAndExit;
435         }
436     }

438     /* If the caller hasn't specified a setup routine, use the default */

440     if (!Setup)
441     {
442         Setup = AcpiEvDefaultRegionSetup;
443     }

445     /* Check for an existing internal object */

447     ObjDesc = AcpiNsGetAttachedObject (Node);
448     if (ObjDesc)
449     {
450         /*
451          * The attached device object already exists. Make sure the handler
452          * is not already installed.
453          */
454         HandlerObj = ObjDesc->Device.Handler;

456         /* Walk the handler list for this device */
```



```
458         while (HandlerObj)
459         {
460             /* Same SpaceId indicates a handler already installed */

462             if (HandlerObj->AddressSpace.SpaceId == SpaceId)
463             {
464                 if (HandlerObj->AddressSpace.Handler == Handler)
465                 {
466                     /*
467                      * It is (relatively) OK to attempt to install the SAME
468                      * handler twice. This can easily happen with the
469                      * PCI_Config space.
470                      */
471                     Status = AE_SAME_HANDLER;
472                     goto UnlockAndExit;
473                 }
474                 else
475                 {
476                     /* A handler is already installed */

478                     Status = AE_ALREADY_EXISTS;
479                 }
480                 goto UnlockAndExit;
481             }

483             /* Walk the linked list of handlers */

485             HandlerObj = HandlerObj->AddressSpace.Next;
486         }
487     }
488     else
489     {
490         ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
491             "Creating object on Device %p while installing handler\n", Node));

493         /* ObjDesc does not exist, create one */

495         if (Node->Type == ACPI_TYPE_ANY)
496         {
497             Type = ACPI_TYPE_DEVICE;
498         }
499         else
500         {
501             Type = Node->Type;
502         }

504         ObjDesc = AcpiUtCreateInternalObject (Type);
505         if (!ObjDesc)
506         {
507             Status = AE_NO_MEMORY;
508             goto UnlockAndExit;
509         }

511         /* Init new descriptor */

513         ObjDesc->Common.Type = (UINT8) Type;

515         /* Attach the new object to the Node */

517         Status = AcpiNsAttachObject (Node, ObjDesc, Type);

519         /* Remove local reference to the object */

521         AcpiUtRemoveReference (ObjDesc);

523         if (ACPI_FAILURE (Status))
```

```
524         {
525             goto UnlockAndExit;
526         }
527     }

529     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
530         "Installing address handler for region %s(%X) on Device %4.4s %p(%p)\n",
531         AcpiUtGetRegionName (SpaceId), SpaceId,
532         AcpiUtGetNodeName (Node), Node, ObjDesc));

534     /*
535      * Install the handler
536      *
537      * At this point there is no existing handler. Just allocate the object
538      * for the handler and link it into the list.
539      */
540     HandlerObj = AcpiUtCreateInternalObject (ACPI_TYPE_LOCAL_ADDRESS_HANDLER);
541     if (!HandlerObj)
542     {
543         Status = AE_NO_MEMORY;
544         goto UnlockAndExit;
545     }

547     /* Init handler obj */

549     HandlerObj->AddressSpace.SpaceId = (UINT8) SpaceId;
550     HandlerObj->AddressSpace.HandlerFlags = Flags;
551     HandlerObj->AddressSpace.RegionList = NULL;
552     HandlerObj->AddressSpace.Node = Node;
553     HandlerObj->AddressSpace.Handler = Handler;
554     HandlerObj->AddressSpace.Context = Context;
555     HandlerObj->AddressSpace.Setup  = Setup;

557     /* Install at head of Device.AddressSpace list */

559     HandlerObj->AddressSpace.Next = ObjDesc->Device.Handler;

561     /*
562      * The Device object is the first reference on the HandlerObj.
563      * Each region that uses the handler adds a reference.
564      */
565     ObjDesc->Device.Handler = HandlerObj;

567     /*
568      * Walk the namespace finding all of the regions this
569      * handler will manage.
570      *
571      * Start at the device and search the branch toward
572      * the leaf nodes until either the leaf is encountered or
573      * a device is detected that has an address handler of the
574      * same type.
575      *
576      * In either case, back up and search down the remainder
577      * of the branch
578      */
579     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, Node, ACPI_UINT32_MAX,
580                 ACPI_NS_WALK_UNLOCK, AcpiEvInstallHandler, NULL,
581                 HandlerObj, NULL);

583 UnlockAndExit:
584     return_ACPI_STATUS (Status);
585 }
```

```
**********************************************************
    9419 Thu Dec 26 13:48:58 2013
new/usr/src/common/acpica/components/events/evmisc.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evmisc - Miscellaneous event manager support functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evmisc")


/* Local prototypes */

static void ACPI_SYSTEM_XFACE
AcpiEvNotifyDispatch (
    void                    *Context);


/******************************************************************************
```



```
 *
 * FUNCTION:    AcpiEvIsNotifyObject
 *
 * PARAMETERS:  Node            - Node to check
 *
 * RETURN:      TRUE if notifies allowed on this object
 *
 * DESCRIPTION: Check type of node for a object that supports notifies.
 *
 *              TBD: This could be replaced by a flag bit in the node.
 *
 *****************************************************************************/

BOOLEAN
AcpiEvIsNotifyObject (
    ACPI_NAMESPACE_NODE     *Node)
{
    switch (Node->Type)
    {
    case ACPI_TYPE_DEVICE:
    case ACPI_TYPE_PROCESSOR:
    case ACPI_TYPE_THERMAL:
        /*
         * These are the ONLY objects that can receive ACPI notifications
         */
        return (TRUE);

    default:

        return (FALSE);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvQueueNotifyRequest
 *
 * PARAMETERS:  Node            - NS node for the notified object
 *              NotifyValue     - Value from the Notify() request
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Dispatch a device notification event to a previously
 *              installed handler.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvQueueNotifyRequest (
    ACPI_NAMESPACE_NODE     *Node,
    UINT32                  NotifyValue)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     *HandlerListHead = NULL;
    ACPI_GENERIC_STATE      *Info;
    UINT8                   HandlerListId = 0;
    ACPI_STATUS             Status = AE_OK;


    ACPI_FUNCTION_NAME (EvQueueNotifyRequest);


    /* Are Notifies allowed on this object? */

    if (!AcpiEvIsNotifyObject (Node))
```

```
    {
        return (AE_TYPE);
    }

    /* Get the correct notify list type (System or Device) */

    if (NotifyValue <= ACPI_MAX_SYS_NOTIFY)
    {
        HandlerListId = ACPI_SYSTEM_HANDLER_LIST;
    }
    else
    {
        HandlerListId = ACPI_DEVICE_HANDLER_LIST;
    }

    /* Get the notify object attached to the namespace Node */

    ObjDesc = AcpiNsGetAttachedObject (Node);
    if (ObjDesc)
    {
        /* We have an attached object, Get the correct handler list */

        HandlerListHead = ObjDesc->CommonNotify.NotifyList[HandlerListId];
    }

    /*
     * If there is no notify handler (Global or Local)
     * for this object, just ignore the notify
     */
    if (!AcpiGbl_GlobalNotify[HandlerListId].Handler && !HandlerListHead)
    {
        ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
            "No notify handler for Notify, ignoring (%4.4s, %X) node %p\n",
            AcpiUtGetNodeName (Node), NotifyValue, Node));

        return (AE_OK);
    }

    /* Setup notify info and schedule the notify dispatcher */

    Info = AcpiUtCreateGenericState ();
    if (!Info)
    {
        return (AE_NO_MEMORY);
    }

    Info->Common.DescriptorType = ACPI_DESC_TYPE_STATE_NOTIFY;

    Info->Notify.Node = Node;
    Info->Notify.Value = (UINT16) NotifyValue;
    Info->Notify.HandlerListId = HandlerListId;
    Info->Notify.HandlerListHead = HandlerListHead;
    Info->Notify.Global = &AcpiGbl_GlobalNotify[HandlerListId];

    ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
        "Dispatching Notify on [%4.4s] (%s) Value 0x%2.2X (%s) Node %p\n",
        AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type),
        NotifyValue, AcpiUtGetNotifyName (NotifyValue), Node));

    Status = AcpiOsExecute (OSL_NOTIFY_HANDLER, AcpiEvNotifyDispatch,
        Info);
    if (ACPI_FAILURE (Status))
    {
        AcpiUtDeleteGenericState (Info);
    }
```



```
    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvNotifyDispatch
 *
 * PARAMETERS:  Context         - To be passed to the notify handler
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Dispatch a device notification event to a previously
 *              installed handler.
 *
 *****************************************************************************/

static void ACPI_SYSTEM_XFACE
AcpiEvNotifyDispatch (
    void                    *Context)
{
    ACPI_GENERIC_STATE      *Info = (ACPI_GENERIC_STATE *) Context;
    ACPI_OPERAND_OBJECT     *HandlerObj;


    ACPI_FUNCTION_ENTRY ();


    /* Invoke a global notify handler if installed */

    if (Info->Notify.Global->Handler)
    {
        Info->Notify.Global->Handler (Info->Notify.Node,
            Info->Notify.Value,
            Info->Notify.Global->Context);
    }

    /* Now invoke the local notify handler(s) if any are installed */

    HandlerObj = Info->Notify.HandlerListHead;
    while (HandlerObj)
    {
        HandlerObj->Notify.Handler (Info->Notify.Node,
            Info->Notify.Value,
            HandlerObj->Notify.Context);

        HandlerObj = HandlerObj->Notify.Next[Info->Notify.HandlerListId];
    }

    /* All done with the info object */

    AcpiUtDeleteGenericState (Info);
}


#if (!ACPI_REDUCED_HARDWARE)
/******************************************************************************
 *
 * FUNCTION:    AcpiEvTerminate
 *
 * PARAMETERS:  none
 *
 * RETURN:      none
 *
 * DESCRIPTION: Disable events and free memory allocated for table storage.
 *
```

```
 *****************************************************************************/

void
AcpiEvTerminate (
    void)
{
    UINT32                  i;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvTerminate);


    if (AcpiGbl_EventsInitialized)
    {
        /*
         * Disable all event-related functionality. In all cases, on error,
         * print a message but obviously we don't abort.
         */

        /* Disable all fixed events */

        for (i = 0; i < ACPI_NUM_FIXED_EVENTS; i++)
        {
            Status = AcpiDisableEvent (i, 0);
            if (ACPI_FAILURE (Status))
            {
                ACPI_ERROR ((AE_INFO,
                    "Could not disable fixed event %u", (UINT32) i));
            }
        }

        /* Disable all GPEs in all GPE blocks */

        Status = AcpiEvWalkGpeList (AcpiHwDisableGpeBlock, NULL);

        Status = AcpiEvRemoveGlobalLockHandler ();
        if (ACPI_FAILURE(Status))
        {
            ACPI_ERROR ((AE_INFO,
                "Could not remove Global Lock handler"));
        }

        AcpiGbl_EventsInitialized = FALSE;
    }

    /* Remove SCI handlers */

    Status = AcpiEvRemoveAllSciHandlers ();
    if (ACPI_FAILURE(Status))
    {
        ACPI_ERROR ((AE_INFO,
            "Could not remove SCI handler"));
    }

    /* Deallocate all handler objects installed within GPE info structs */

    Status = AcpiEvWalkGpeList (AcpiEvDeleteGpeHandlers, NULL);

    /* Return to original mode if necessary */

    if (AcpiGbl_OriginalMode == ACPI_SYS_MODE_LEGACY)
    {
        Status = AcpiDisable ();
        if (ACPI_FAILURE (Status))
        {
```



```
            ACPI_WARNING ((AE_INFO, "AcpiDisable failed"));
        }
    }
    return_VOID;
}

#endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   24892 Thu Dec 26 13:48:58 2013
new/usr/src/common/acpica/components/events/evregion.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evregion - Operation Region support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EVREGION_C__

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evregion")


extern UINT8        AcpiGbl_DefaultAddressSpaces[];

/* Local prototypes */
```



```
static void
AcpiEvOrphanEcRegMethod (
    ACPI_NAMESPACE_NODE     *EcDeviceNode);

static ACPI_STATUS
AcpiEvRegRun (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    AcpiEvInitializeOpRegions
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute _REG methods for all Operation Regions that have
 *              an installed default region handler.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvInitializeOpRegions (
    void)
{
    ACPI_STATUS             Status;
    UINT32                  i;


    ACPI_FUNCTION_TRACE (EvInitializeOpRegions);


    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Run the _REG methods for OpRegions in each default address space */

    for (i = 0; i < ACPI_NUM_DEFAULT_SPACES; i++)
    {
        /*
         * Make sure the installed handler is the DEFAULT handler. If not the
         * default, the _REG methods will have already been run (when the
         * handler was installed)
         */
        if (AcpiEvHasDefaultHandler (AcpiGbl_RootNode,
                AcpiGbl_DefaultAddressSpaces[i]))
        {
            Status = AcpiEvExecuteRegMethods (AcpiGbl_RootNode,
                        AcpiGbl_DefaultAddressSpaces[i]);
        }
    }

    AcpiGbl_RegMethodsExecuted = TRUE;

    (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    return_ACPI_STATUS (Status);
}
```

```
127 /*******************************************************************************
128  *
129  * FUNCTION:    AcpiEvAddressSpaceDispatch
130  *
131  * PARAMETERS:  RegionObj           - Internal region object
132  *              FieldObj            - Corresponding field. Can be NULL.
133  *              Function            - Read or Write operation
134  *              RegionOffset        - Where in the region to read or write
135  *              BitWidth            - Field width in bits (8, 16, 32, or 64)
136  *              Value               - Pointer to in or out value, must be
137  *                                    a full 64-bit integer
138  *
139  * RETURN:      Status
140  *
141  * DESCRIPTION: Dispatch an address space or operation region access to
142  *              a previously installed handler.
143  *
144  ******************************************************************************/

146 ACPI_STATUS
147 AcpiEvAddressSpaceDispatch (
148     ACPI_OPERAND_OBJECT     *RegionObj,
149     ACPI_OPERAND_OBJECT     *FieldObj,
150     UINT32                  Function,
151     UINT32                  RegionOffset,
152     UINT32                  BitWidth,
153     UINT64                  *Value)
154 {
155     ACPI_STATUS             Status;
156     ACPI_ADR_SPACE_HANDLER  Handler;
157     ACPI_ADR_SPACE_SETUP    RegionSetup;
158     ACPI_OPERAND_OBJECT     *HandlerDesc;
159     ACPI_OPERAND_OBJECT     *RegionObj2;
160     void                    *RegionContext = NULL;
161     ACPI_CONNECTION_INFO    *Context;


164     ACPI_FUNCTION_TRACE (EvAddressSpaceDispatch);


167     RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
168     if (!RegionObj2)
169     {
170         return_ACPI_STATUS (AE_NOT_EXIST);
171     }

173     /* Ensure that there is a handler associated with this region */

175     HandlerDesc = RegionObj->Region.Handler;
176     if (!HandlerDesc)
177     {
178         ACPI_ERROR ((AE_INFO,
179             "No handler for Region [%4.4s] (%p) [%s]",
180             AcpiUtGetNodeName (RegionObj->Region.Node),
181             RegionObj, AcpiUtGetRegionName (RegionObj->Region.SpaceId)));

183         return_ACPI_STATUS (AE_NOT_EXIST);
184     }

186     Context = HandlerDesc->AddressSpace.Context;

188     /*
189      * It may be the case that the region has never been initialized.
190      * Some types of regions require special init code
191      */
192     if (!(RegionObj->Region.Flags & AOPOBJ_SETUP_COMPLETE))
```



```
193     {
194         /* This region has not been initialized yet, do it */

196         RegionSetup = HandlerDesc->AddressSpace.Setup;
197         if (!RegionSetup)
198         {
199             /* No initialization routine, exit with error */

201             ACPI_ERROR ((AE_INFO,
202                 "No init routine for region(%p) [%s]",
203                 RegionObj, AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
204             return_ACPI_STATUS (AE_NOT_EXIST);
205         }

207         /*
208          * We must exit the interpreter because the region setup will
209          * potentially execute control methods (for example, the _REG method
210          * for this region)
211          */
212         AcpiExExitInterpreter ();

214         Status = RegionSetup (RegionObj, ACPI_REGION_ACTIVATE,
215                     Context, &RegionContext);

217         /* Re-enter the interpreter */

219         AcpiExEnterInterpreter ();

221         /* Check for failure of the Region Setup */

223         if (ACPI_FAILURE (Status))
224         {
225             ACPI_EXCEPTION ((AE_INFO, Status,
226                 "During region initialization: [%s]",
227                 AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
228             return_ACPI_STATUS (Status);
229         }

231         /* Region initialization may have been completed by RegionSetup */

233         if (!(RegionObj->Region.Flags & AOPOBJ_SETUP_COMPLETE))
234         {
235             RegionObj->Region.Flags |= AOPOBJ_SETUP_COMPLETE;

237             /*
238              * Save the returned context for use in all accesses to
239              * the handler for this particular region
240              */
241             if (!(RegionObj2->Extra.RegionContext))
242             {
243                 RegionObj2->Extra.RegionContext = RegionContext;
244             }
245         }
246     }

248     /* We have everything we need, we can invoke the address space handler */

250     Handler = HandlerDesc->AddressSpace.Handler;

252     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
253         "Handler %p (@%p) Address %8.8X%8.8X [%s]\n",
254         &RegionObj->Region.Handler->AddressSpace, Handler,
255         ACPI_FORMAT_NATIVE_UINT (RegionObj->Region.Address + RegionOffset),
256         AcpiUtGetRegionName (RegionObj->Region.SpaceId)));

258     /*
```

```
259      * Special handling for GenericSerialBus and GeneralPurposeIo:
260      * There are three extra parameters that must be passed to the
261      * handler via the context:
262      *   1) Connection buffer, a resource template from Connection() op.
263      *   2) Length of the above buffer.
264      *   3) Actual access length from the AccessAs() op.
265      */
266     if (((RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS) ||
267             (RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GPIO)) &&
268         Context &&
269         FieldObj)
270     {
271         /* Get the Connection (ResourceTemplate) buffer */

273         Context->Connection = FieldObj->Field.ResourceBuffer;
274         Context->Length = FieldObj->Field.ResourceLength;
275         Context->AccessLength = FieldObj->Field.AccessLength;
276     }

278     if (!(HandlerDesc->AddressSpace.HandlerFlags &
279             ACPI_ADDR_HANDLER_DEFAULT_INSTALLED))
280     {
281         /*
282          * For handlers other than the default (supplied) handlers, we must
283          * exit the interpreter because the handler *might* block -- we don't
284          * know what it will do, so we can't hold the lock on the intepreter.
285          */
286         AcpiExExitInterpreter();
287     }

289     /* Call the handler */

291     Status = Handler (Function,
292         (RegionObj->Region.Address + RegionOffset), BitWidth, Value,
293         Context, RegionObj2->Extra.RegionContext);

295     if (ACPI_FAILURE (Status))
296     {
297         ACPI_EXCEPTION ((AE_INFO, Status, "Returned by Handler for [%s]",
298             AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
299     }

301     if (!(HandlerDesc->AddressSpace.HandlerFlags &
302             ACPI_ADDR_HANDLER_DEFAULT_INSTALLED))
303     {
304         /*
305          * We just returned from a non-default handler, we must re-enter the
306          * interpreter
307          */
308        AcpiExEnterInterpreter ();
309     }

311     return_ACPI_STATUS (Status);
312 }


315 /*******************************************************************************
316  *
317  * FUNCTION:    AcpiEvDetachRegion
318  *
319  * PARAMETERS:  RegionObj           - Region Object
320  *              AcpiNsIsLocked      - Namespace Region Already Locked?
321  *
322  * RETURN:      None
323  *
324  * DESCRIPTION: Break the association between the handler and the region
```



```
325  *              this is a two way association.
326  *
327  ******************************************************************************/

329 void
330 AcpiEvDetachRegion(
331     ACPI_OPERAND_OBJECT     *RegionObj,
332     BOOLEAN                 AcpiNsIsLocked)
333 {
334     ACPI_OPERAND_OBJECT     *HandlerObj;
335     ACPI_OPERAND_OBJECT     *ObjDesc;
336     ACPI_OPERAND_OBJECT     **LastObjPtr;
337     ACPI_ADR_SPACE_SETUP    RegionSetup;
338     void                    **RegionContext;
339     ACPI_OPERAND_OBJECT     *RegionObj2;
340     ACPI_STATUS             Status;


343     ACPI_FUNCTION_TRACE (EvDetachRegion);


346     RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
347     if (!RegionObj2)
348     {
349         return_VOID;
350     }
351     RegionContext = &RegionObj2->Extra.RegionContext;

353     /* Get the address handler from the region object */

355     HandlerObj = RegionObj->Region.Handler;
356     if (!HandlerObj)
357     {
358         /* This region has no handler, all done */

360         return_VOID;
361     }

363     /* Find this region in the handler's list */

365     ObjDesc = HandlerObj->AddressSpace.RegionList;
366     LastObjPtr = &HandlerObj->AddressSpace.RegionList;

368     while (ObjDesc)
369     {
370         /* Is this the correct Region? */

372         if (ObjDesc == RegionObj)
373         {
374             ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
375                 "Removing Region %p from address handler %p\n",
376                 RegionObj, HandlerObj));

378             /* This is it, remove it from the handler's list */

380             *LastObjPtr = ObjDesc->Region.Next;
381             ObjDesc->Region.Next = NULL;        /* Must clear field */

383             if (AcpiNsIsLocked)
384             {
385                 Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
386                 if (ACPI_FAILURE (Status))
387                 {
388                     return_VOID;
389                 }
390             }
```

```
392             /* Now stop region accesses by executing the _REG method */
394             Status = AcpiEvExecuteRegMethod (RegionObj, ACPI_REG_DISCONNECT);
395             if (ACPI_FAILURE (Status))
396             {
397                 ACPI_EXCEPTION ((AE_INFO, Status, "from region _REG, [%s]",
398                     AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
399             }
401             if (AcpiNsIsLocked)
402             {
403                 Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
404                 if (ACPI_FAILURE (Status))
405                 {
406                     return_VOID;
407                 }
408             }
410             /*
411              * If the region has been activated, call the setup handler with
412              * the deactivate notification
413              */
414             if (RegionObj->Region.Flags & AOPOBJ_SETUP_COMPLETE)
415             {
416                 RegionSetup = HandlerObj->AddressSpace.Setup;
417                 Status = RegionSetup (RegionObj, ACPI_REGION_DEACTIVATE,
418                     HandlerObj->AddressSpace.Context, RegionContext);
420                 /*
421                  * RegionContext should have been released by the deactivate
422                  * operation. We don't need access to it anymore here.
423                  */
424                 if (RegionContext)
425                 {
426                     *RegionContext = NULL;
427                 }
429                 /* Init routine may fail, Just ignore errors */
431                 if (ACPI_FAILURE (Status))
432                 {
433                     ACPI_EXCEPTION ((AE_INFO, Status,
434                         "from region handler - deactivate, [%s]",
435                         AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
436                 }
438                 RegionObj->Region.Flags &= ~(AOPOBJ_SETUP_COMPLETE);
439             }
441             /*
442              * Remove handler reference in the region
443              *
444              * NOTE: this doesn't mean that the region goes away, the region
445              * is just inaccessible as indicated to the _REG method
446              *
447              * If the region is on the handler's list, this must be the
448              * region's handler
449              */
450             RegionObj->Region.Handler = NULL;
451             AcpiUtRemoveReference (HandlerObj);
453             return_VOID;
454         }
456         /* Walk the linked list of handlers */
```



```
458         LastObjPtr = &ObjDesc->Region.Next;
459         ObjDesc = ObjDesc->Region.Next;
460     }
462     /* If we get here, the region was not in the handler's region list */
464     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
465         "Cannot remove region %p from address handler %p\n",
466         RegionObj, HandlerObj));
468     return_VOID;
469 }
472 /*******************************************************************************
473  *
474  * FUNCTION:    AcpiEvAttachRegion
475  *
476  * PARAMETERS:  HandlerObj          - Handler Object
477  *              RegionObj           - Region Object
478  *              AcpiNsIsLocked      - Namespace Region Already Locked?
479  *
480  * RETURN:      None
481  *
482  * DESCRIPTION: Create the association between the handler and the region
483  *              this is a two way association.
484  *
485  ******************************************************************************/
487 ACPI_STATUS
488 AcpiEvAttachRegion (
489     ACPI_OPERAND_OBJECT     *HandlerObj,
490     ACPI_OPERAND_OBJECT     *RegionObj,
491     BOOLEAN                 AcpiNsIsLocked)
492 {
494     ACPI_FUNCTION_TRACE (EvAttachRegion);
497     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
498         "Adding Region [%4.4s] %p to address handler %p [%s]\n",
499         AcpiUtGetNodeName (RegionObj->Region.Node),
500         RegionObj, HandlerObj,
501         AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
503     /* Link this region to the front of the handler's list */
505     RegionObj->Region.Next = HandlerObj->AddressSpace.RegionList;
506     HandlerObj->AddressSpace.RegionList = RegionObj;
508     /* Install the region's handler */
510     if (RegionObj->Region.Handler)
511     {
512         return_ACPI_STATUS (AE_ALREADY_EXISTS);
513     }
515     RegionObj->Region.Handler = HandlerObj;
516     AcpiUtAddReference (HandlerObj);
518     return_ACPI_STATUS (AE_OK);
519 }
522 /*******************************************************************************
```

```
 *
 * FUNCTION:    AcpiEvExecuteRegMethod
 *
 * PARAMETERS:  RegionObj           - Region object
 *              Function            - Passed to _REG: On (1) or Off (0)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute _REG method for a region
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEvExecuteRegMethod (
    ACPI_OPERAND_OBJECT     *RegionObj,
    UINT32                  Function)
{
    ACPI_EVALUATE_INFO      *Info;
    ACPI_OPERAND_OBJECT     *Args[3];
    ACPI_OPERAND_OBJECT     *RegionObj2;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvExecuteRegMethod);


    RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
    if (!RegionObj2)
    {
        return_ACPI_STATUS (AE_NOT_EXIST);
    }

    if (RegionObj2->Extra.Method_REG == NULL)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /* Allocate and initialize the evaluation information block */

    Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
    if (!Info)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    Info->PrefixNode = RegionObj2->Extra.Method_REG;
    Info->RelativePathname = NULL;
    Info->Parameters = Args;
    Info->Flags = ACPI_IGNORE_RETURN_VALUE;

    /*
     * The _REG method has two arguments:
     *
     * Arg0 - Integer:
     *  Operation region space ID Same value as RegionObj->Region.SpaceId
     *
     * Arg1 - Integer:
     *  connection status 1 for connecting the handler, 0 for disconnecting
     *  the handler (Passed as a parameter)
     */
    Args[0] = AcpiUtCreateIntegerObject ((UINT64) RegionObj->Region.SpaceId);
    if (!Args[0])
    {
        Status = AE_NO_MEMORY;
        goto Cleanup1;
    }
```



```
    Args[1] = AcpiUtCreateIntegerObject ((UINT64) Function);
    if (!Args[1])
    {
        Status = AE_NO_MEMORY;
        goto Cleanup2;
    }

    Args[2] = NULL; /* Terminate list */

    /* Execute the method, no return value */

    ACPI_DEBUG_EXEC (
        AcpiUtDisplayInitPathname (ACPI_TYPE_METHOD, Info->PrefixNode, NULL));

    Status = AcpiNsEvaluate (Info);
    AcpiUtRemoveReference (Args[1]);

Cleanup2:
    AcpiUtRemoveReference (Args[0]);

Cleanup1:
    ACPI_FREE (Info);
    return_ACPI_STATUS (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvExecuteRegMethods
 *
 * PARAMETERS:  Node            - Namespace node for the device
 *              SpaceId         - The address space ID
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Run all _REG methods for the input Space ID;
 *              Note: assumes namespace is locked, or system init time.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvExecuteRegMethods (
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_ADR_SPACE_TYPE     SpaceId)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (EvExecuteRegMethods);


    /*
     * Run all _REG methods for all Operation Regions for this space ID. This
     * is a separate walk in order to handle any interdependencies between
     * regions and _REG methods. (i.e. handlers must be installed for all
     * regions of this Space ID before we can run any _REG methods)
     */
    Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, Node, ACPI_UINT32_MAX,
                ACPI_NS_WALK_UNLOCK, AcpiEvRegRun, NULL,
                &SpaceId, NULL);

    /* Special case for EC: handle "orphan" _REG methods with no region */

    if (SpaceId == ACPI_ADR_SPACE_EC)
    {
```

```
655         AcpiEvOrphanEcRegMethod (Node);
656     }

658     return_ACPI_STATUS (Status);
659 }


662 /*******************************************************************************
663  *
664  * FUNCTION:    AcpiEvRegRun
665  *
666  * PARAMETERS:  WalkNamespace callback
667  *
668  * DESCRIPTION: Run _REG method for region objects of the requested spaceID
669  *
670  ******************************************************************************/

672 static ACPI_STATUS
673 AcpiEvRegRun (
674     ACPI_HANDLE             ObjHandle,
675     UINT32                  Level,
676     void                    *Context,
677     void                    **ReturnValue)
678 {
679     ACPI_OPERAND_OBJECT     *ObjDesc;
680     ACPI_NAMESPACE_NODE     *Node;
681     ACPI_ADR_SPACE_TYPE     SpaceId;
682     ACPI_STATUS             Status;


685     SpaceId = *ACPI_CAST_PTR (ACPI_ADR_SPACE_TYPE, Context);

687     /* Convert and validate the device handle */

689     Node = AcpiNsValidateHandle (ObjHandle);
690     if (!Node)
691     {
692         return (AE_BAD_PARAMETER);
693     }

695     /*
696      * We only care about regions.and objects that are allowed to have address
697      * space handlers
698      */
699     if ((Node->Type != ACPI_TYPE_REGION) &&
700         (Node != AcpiGbl_RootNode))
701     {
702         return (AE_OK);
703     }

705     /* Check for an existing internal object */

707     ObjDesc = AcpiNsGetAttachedObject (Node);
708     if (!ObjDesc)
709     {
710         /* No object, just exit */

712         return (AE_OK);
713     }

715     /* Object is a Region */

717     if (ObjDesc->Region.SpaceId != SpaceId)
718     {
719         /* This region is for a different address space, just ignore it */
```



```
721         return (AE_OK);
722     }

724     Status = AcpiEvExecuteRegMethod (ObjDesc, ACPI_REG_CONNECT);
725     return (Status);
726 }


729 /*******************************************************************************
730  *
731  * FUNCTION:    AcpiEvOrphanEcRegMethod
732  *
733  * PARAMETERS:  EcDeviceNode        - Namespace node for an EC device
734  *
735  * RETURN:      None
736  *
737  * DESCRIPTION: Execute an "orphan" _REG method that appears under the EC
738  *              device. This is a _REG method that has no corresponding region
739  *              within the EC device scope. The orphan _REG method appears to
740  *              have been enabled by the description of the ECDT in the ACPI
741  *              specification: "The availability of the region space can be
742  *              detected by providing a _REG method object underneath the
743  *              Embedded Controller device."
744  *
745  *              To quickly access the EC device, we use the EcDeviceNode used
746  *              during EC handler installation. Otherwise, we would need to
747  *              perform a time consuming namespace walk, executing _HID
748  *              methods to find the EC device.
749  *
750  *  MUTEX:      Assumes the namespace is locked
751  *
752  ******************************************************************************/

754 static void
755 AcpiEvOrphanEcRegMethod (
756     ACPI_NAMESPACE_NODE     *EcDeviceNode)
757 {
758     ACPI_HANDLE             RegMethod;
759     ACPI_NAMESPACE_NODE     *NextNode;
760     ACPI_STATUS             Status;
761     ACPI_OBJECT_LIST        Args;
762     ACPI_OBJECT             Objects[2];


765     ACPI_FUNCTION_TRACE (EvOrphanEcRegMethod);


768     if (!EcDeviceNode)
769     {
770         return_VOID;
771     }

773     /* Namespace is currently locked, must release */

775     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

777     /* Get a handle to a _REG method immediately under the EC device */

779     Status = AcpiGetHandle (EcDeviceNode, METHOD_NAME__REG, &RegMethod);
780     if (ACPI_FAILURE (Status))
781     {
782         goto Exit; /* There is no _REG method present */
783     }

785     /*
786      * Execute the _REG method only if there is no Operation Region in
```

```
787      * this scope with the Embedded Controller space ID. Otherwise, it
788      * will already have been executed. Note, this allows for Regions
789      * with other space IDs to be present; but the code below will then
790      * execute the _REG method with the EmbeddedControl SpaceID argument.
791      */
792     NextNode = AcpiNsGetNextNode (EcDeviceNode, NULL);
793     while (NextNode)
794     {
795         if ((NextNode->Type == ACPI_TYPE_REGION) &&
796             (NextNode->Object) &&
797             (NextNode->Object->Region.SpaceId == ACPI_ADR_SPACE_EC))
798         {
799             goto Exit; /* Do not execute the _REG */
800         }

802         NextNode = AcpiNsGetNextNode (EcDeviceNode, NextNode);
803     }

805     /* Evaluate the _REG(EmbeddedControl,Connect) method */

807     Args.Count = 2;
808     Args.Pointer = Objects;
809     Objects[0].Type = ACPI_TYPE_INTEGER;
810     Objects[0].Integer.Value = ACPI_ADR_SPACE_EC;
811     Objects[1].Type = ACPI_TYPE_INTEGER;
812     Objects[1].Integer.Value = ACPI_REG_CONNECT;

814     Status = AcpiEvaluateObject (RegMethod, NULL, &Args, NULL);

816 Exit:
817     /* We ignore all errors from above, don't care */

819     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
820     return_VOID;
821 }
```

****
22020 Thu Dec 26 13:48:58 2013
new/usr/src/common/acpica/components/events/evrgnini.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: evrgnini- ACPI AddressSpace (OpRegion) init
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EVRGNINI_C__

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evrgnini")

/* Local prototypes */

static BOOLEAN
AcpiEvIsPciRootBridge (
    ACPI_NAMESPACE_NODE     *Node);
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiEvSystemMemoryRegionSetup
 *
 * PARAMETERS:  Handle              - Region we are interested in
 *              Function            - Start or stop
 *              HandlerContext      - Address space handler context
 *              RegionContext       - Region specific context
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Setup a SystemMemory operation region
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEvSystemMemoryRegionSetup (
    ACPI_HANDLE             Handle,
    UINT32                  Function,
    void                    *HandlerContext,
    void                    **RegionContext)
{
    ACPI_OPERAND_OBJECT     *RegionDesc = (ACPI_OPERAND_OBJECT *) Handle;
    ACPI_MEM_SPACE_CONTEXT  *LocalRegionContext;


    ACPI_FUNCTION_TRACE (EvSystemMemoryRegionSetup);


    if (Function == ACPI_REGION_DEACTIVATE)
    {
        if (*RegionContext)
        {
            LocalRegionContext = (ACPI_MEM_SPACE_CONTEXT *) *RegionContext;

            /* Delete a cached mapping if present */

            if (LocalRegionContext->MappedLength)
            {
                AcpiOsUnmapMemory (LocalRegionContext->MappedLogicalAddress,
                    LocalRegionContext->MappedLength);
            }
            ACPI_FREE (LocalRegionContext);
            *RegionContext = NULL;
        }
        return_ACPI_STATUS (AE_OK);
    }

    /* Create a new context */

    LocalRegionContext = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_MEM_SPACE_CONTEXT));
    if (!(LocalRegionContext))
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    /* Save the region length and address for use in the handler */

    LocalRegionContext->Length  = RegionDesc->Region.Length;
    LocalRegionContext->Address = RegionDesc->Region.Address;

    *RegionContext = LocalRegionContext;
    return_ACPI_STATUS (AE_OK);
}
```

_____unchanged_portion_omitted_

```
374 /*******************************************************************************
375  *
376  * FUNCTION:    AcpiEvIsPciRootBridge
377  *
378  * PARAMETERS:  Node            - Device node being examined
379  *
380  * RETURN:      TRUE if device is a PCI/PCI-Express Root Bridge
381  *
382  * DESCRIPTION: Determine if the input device represents a PCI Root Bridge by
383  *              examining the _HID and _CID for the device.
384  *
385  ******************************************************************************/

387 static BOOLEAN
388 AcpiEvIsPciRootBridge (
389     ACPI_NAMESPACE_NODE     *Node)
390 {
391     ACPI_STATUS             Status;
392     ACPI_PNP_DEVICE_ID      *Hid;
393     ACPI_PNP_DEVICE_ID_LIST *Cid;
392     ACPI_DEVICE_ID          *Hid;
393     ACPI_DEVICE_ID_LIST     *Cid;
394     UINT32                  i;
395     BOOLEAN                 Match;


398     /* Get the _HID and check for a PCI Root Bridge */

400     Status = AcpiUtExecute_HID (Node, &Hid);
401     if (ACPI_FAILURE (Status))
402     {
403         return (FALSE);
404     }

406     Match = AcpiUtIsPciRootBridge (Hid->String);
407     ACPI_FREE (Hid);

409     if (Match)
410     {
411         return (TRUE);
412     }

414     /* The _HID did not match. Get the _CID and check for a PCI Root Bridge */

416     Status = AcpiUtExecute_CID (Node, &Cid);
417     if (ACPI_FAILURE (Status))
418     {
419         return (FALSE);
420     }

422     /* Check all _CIDs in the returned list */

424     for (i = 0; i < Cid->Count; i++)
425     {
426         if (AcpiUtIsPciRootBridge (Cid->Ids[i].String))
427         {
428             ACPI_FREE (Cid);
429             return (TRUE);
430         }
431     }

433     ACPI_FREE (Cid);
434     return (FALSE);
435 }
```

*unchanged_portion_omitted_*


```
538 /*******************************************************************************
539  *
540  * FUNCTION:    AcpiEvInitializeRegion
541  *
542  * PARAMETERS:  RegionObj       - Region we are initializing
543  *              AcpiNsLocked    - Is namespace locked?
544  *
545  * RETURN:      Status
546  *
547  * DESCRIPTION: Initializes the region, finds any _REG methods and saves them
548  *              for execution at a later time
549  *
550  *              Get the appropriate address space handler for a newly
551  *              created region.
552  *
553  *              This also performs address space specific initialization. For
554  *              example, PCI regions must have an _ADR object that contains
555  *              a PCI address in the scope of the definition. This address is
556  *              required to perform an access to PCI config space.
557  *
558  * MUTEX:       Interpreter should be unlocked, because we may run the _REG
559  *              method for this region.
560  *
561  ******************************************************************************/

563 ACPI_STATUS
564 AcpiEvInitializeRegion (
565     ACPI_OPERAND_OBJECT     *RegionObj,
566     BOOLEAN                 AcpiNsLocked)
567 {
568     ACPI_OPERAND_OBJECT     *HandlerObj;
569     ACPI_OPERAND_OBJECT     *ObjDesc;
570     ACPI_ADR_SPACE_TYPE     SpaceId;
571     ACPI_NAMESPACE_NODE     *Node;
572     ACPI_STATUS             Status;
573     ACPI_NAMESPACE_NODE     *MethodNode;
574     ACPI_NAME               *RegNamePtr = (ACPI_NAME *) METHOD_NAME__REG;
575     ACPI_OPERAND_OBJECT     *RegionObj2;


578     ACPI_FUNCTION_TRACE_U32 (EvInitializeRegion, AcpiNsLocked);


581     if (!RegionObj)
582     {
583         return_ACPI_STATUS (AE_BAD_PARAMETER);
584     }

586     if (RegionObj->Common.Flags & AOPOBJ_OBJECT_INITIALIZED)
587     {
588         return_ACPI_STATUS (AE_OK);
589     }

591     RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
592     if (!RegionObj2)
593     {
594         return_ACPI_STATUS (AE_NOT_EXIST);
595     }

597     Node = RegionObj->Region.Node->Parent;
598     SpaceId = RegionObj->Region.SpaceId;

600     /* Setup defaults */
```

```
602     RegionObj->Region.Handler = NULL;
603     RegionObj2->Extra.Method_REG = NULL;
604     RegionObj->Common.Flags &= ~(AOPOBJ_SETUP_COMPLETE);
605     RegionObj->Common.Flags |= AOPOBJ_OBJECT_INITIALIZED;

607     /* Find any "_REG" method associated with this region definition */

609     Status = AcpiNsSearchOneScope (
610                 *RegNamePtr, Node, ACPI_TYPE_METHOD, &MethodNode);
611     if (ACPI_SUCCESS (Status))
612     {
613         /*
614          * The _REG method is optional and there can be only one per region
615          * definition. This will be executed when the handler is attached
616          * or removed
617          */
618         RegionObj2->Extra.Method_REG = MethodNode;
619     }

621     /*
622      * The following loop depends upon the root Node having no parent
623      * ie: AcpiGbl_RootNode->ParentEntry being set to NULL
624      */
625     while (Node)
626     {
627         /* Check to see if a handler exists */

629         HandlerObj = NULL;
630         ObjDesc = AcpiNsGetAttachedObject (Node);
631         if (ObjDesc)
632         {
633             /* Can only be a handler if the object exists */

635             switch (Node->Type)
636             {
637             case ACPI_TYPE_DEVICE:

639                 HandlerObj = ObjDesc->Device.Handler;
640                 break;

642             case ACPI_TYPE_PROCESSOR:

644                 HandlerObj = ObjDesc->Processor.Handler;
645                 break;

647             case ACPI_TYPE_THERMAL:

649                 HandlerObj = ObjDesc->ThermalZone.Handler;
650                 break;

652             case ACPI_TYPE_METHOD:
653                 /*
654                  * If we are executing module level code, the original
655                  * Node's object was replaced by this Method object and we
656                  * saved the handler in the method object.
657                  *
658                  * See AcpiNsExecModuleCode
659                  */
660                 if (ObjDesc->Method.InfoFlags & ACPI_METHOD_MODULE_LEVEL)
661                 {
662                     HandlerObj = ObjDesc->Method.Dispatch.Handler;
663                 }
664                 break;

666             default:
```



```
668                 /* Ignore other objects */

670                 break;
671             }

673             while (HandlerObj)
674             {
675                 /* Is this handler of the correct type? */

677                 if (HandlerObj->AddressSpace.SpaceId == SpaceId)
678                 {
679                     /* Found correct handler */

681                     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
682                         "Found handler %p for region %p in obj %p\n",
683                         HandlerObj, RegionObj, ObjDesc));

685                     Status = AcpiEvAttachRegion (HandlerObj, RegionObj,
686                                 AcpiNsLocked);

688                     /*
689                      * Tell all users that this region is usable by
690                      * running the _REG method
691                      */
692                     if (AcpiNsLocked)
693                     {
694                         Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
695                         if (ACPI_FAILURE (Status))
696                         {
697                             return_ACPI_STATUS (Status);
698                         }
699                     }

701                     Status = AcpiEvExecuteRegMethod (RegionObj, ACPI_REG_CONNECT

703                     if (AcpiNsLocked)
704                     {
705                         Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
706                         if (ACPI_FAILURE (Status))
707                         {
708                             return_ACPI_STATUS (Status);
709                         }
710                     }

712                     return_ACPI_STATUS (AE_OK);
713                 }

715                 /* Try next handler in the list */

717                 HandlerObj = HandlerObj->AddressSpace.Next;
718             }
719         }

721         /* This node does not have the handler we need; Pop up one level */

723         Node = Node->Parent;
724     }

726     /* If we get here, there is no handler for this region */

728     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
729         "No handler for RegionType %s(%X) (RegionObj %p)\n",
730         AcpiUtGetRegionName (SpaceId), SpaceId, RegionObj));

732     return_ACPI_STATUS (AE_NOT_EXIST);
733 }
```

```
**********************************************************
    8237 Thu Dec 26 13:48:59 2013
new/usr/src/common/acpica/components/events/evsci.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evsci - System Control Interrupt configuration and
 *                      legacy to ACPI mode state transition functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"


#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evsci")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/* Local prototypes */

static UINT32 ACPI_SYSTEM_XFACE
AcpiEvSciXruptHandler (
    void                    *Context);
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiEvSciDispatch
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status code indicates whether interrupt was handled.
 *
 * DESCRIPTION: Dispatch the SCI to all host-installed SCI handlers.
 *
 *****************************************************************************/

UINT32
AcpiEvSciDispatch (
    void)
{
    ACPI_SCI_HANDLER_INFO   *SciHandler;
    ACPI_CPU_FLAGS          Flags;
    UINT32                  IntStatus = ACPI_INTERRUPT_NOT_HANDLED;


    ACPI_FUNCTION_NAME (EvSciDispatch);


    /* Are there any host-installed SCI handlers? */

    if (!AcpiGbl_SciHandlerList)
    {
        return (IntStatus);
    }

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

    /* Invoke all host-installed SCI handlers */

    SciHandler = AcpiGbl_SciHandlerList;
    while (SciHandler)
    {
        /* Invoke the installed handler (at interrupt level) */

        IntStatus |= SciHandler->Address (
            SciHandler->Context);

        SciHandler = SciHandler->Next;
    }

    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    return (IntStatus);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiEvSciXruptHandler
 *
 * PARAMETERS:  Context   - Calling Context
 *
 * RETURN:      Status code indicates whether interrupt was handled.
 *
 * DESCRIPTION: Interrupt handler that will figure out what function or
 *              control method to call to deal with a SCI.
 *
 *****************************************************************************/

static UINT32 ACPI_SYSTEM_XFACE
```

```
127 AcpiEvSciXruptHandler (
128     void                    *Context)
129 {
130     ACPI_GPE_XRUPT_INFO     *GpeXruptList = Context;
131     UINT32                  InterruptHandled = ACPI_INTERRUPT_NOT_HANDLED;


134     ACPI_FUNCTION_TRACE (EvSciXruptHandler);


137     /*
138      * We are guaranteed by the ACPI CA initialization/shutdown code that
139      * if this interrupt handler is installed, ACPI is enabled.
140      */

142     /*
143      * Fixed Events:
144      * Check for and dispatch any Fixed Events that have occurred
145      */
146     InterruptHandled |= AcpiEvFixedEventDetect ();

148     /*
149      * General Purpose Events:
150      * Check for and dispatch any GPEs that have occurred
151      */
152     InterruptHandled |= AcpiEvGpeDetect (GpeXruptList);

154     /* Invoke all host-installed SCI handlers */

156     InterruptHandled |= AcpiEvSciDispatch ();

158     AcpiSciCount++;
159     return_UINT32 (InterruptHandled);
160 }


163 /******************************************************************************
164  *
165  * FUNCTION:    AcpiEvGpeXruptHandler
166  *
167  * PARAMETERS:  Context   - Calling Context
168  *
169  * RETURN:      Status code indicates whether interrupt was handled.
170  *
171  * DESCRIPTION: Handler for GPE Block Device interrupts
172  *
173  *****************************************************************************/

175 UINT32 ACPI_SYSTEM_XFACE
176 AcpiEvGpeXruptHandler (
177     void                    *Context)
178 {
179     ACPI_GPE_XRUPT_INFO     *GpeXruptList = Context;
180     UINT32                  InterruptHandled = ACPI_INTERRUPT_NOT_HANDLED;


183     ACPI_FUNCTION_TRACE (EvGpeXruptHandler);


186     /*
187      * We are guaranteed by the ACPICA initialization/shutdown code that
188      * if this interrupt handler is installed, ACPI is enabled.
189      */

191     /* GPEs: Check for and dispatch any GPEs that have occurred */
```



```
193     InterruptHandled |= AcpiEvGpeDetect (GpeXruptList);
194     return_UINT32 (InterruptHandled);
195 }


198 /******************************************************************************
199  *
200  * FUNCTION:    AcpiEvInstallSciHandler
201  *
202  * PARAMETERS:  none
203  *
204  * RETURN:      Status
205  *
206  * DESCRIPTION: Installs SCI handler.
207  *
208  *****************************************************************************/

210 UINT32
211 AcpiEvInstallSciHandler (
212     void)
213 {
214     UINT32                  Status = AE_OK;


217     ACPI_FUNCTION_TRACE (EvInstallSciHandler);


220     Status = AcpiOsInstallInterruptHandler ((UINT32) AcpiGbl_FADT.SciInterrupt,
221                 AcpiEvSciXruptHandler, AcpiGbl_GpeXruptListHead);
222     return_ACPI_STATUS (Status);
223 }


226 /******************************************************************************
227  *
228  * FUNCTION:    AcpiEvRemoveAllSciHandlers
229  *
230  * PARAMETERS:  none
231  *
232  * RETURN:      AE_OK if handler uninstalled, AE_ERROR if handler was not
233  *              installed to begin with
234  *
235  * DESCRIPTION: Remove the SCI interrupt handler. No further SCIs will be
236  *              taken. Remove all host-installed SCI handlers.
237  *
238  * Note:  It doesn't seem important to disable all events or set the event
239  *        enable registers to their original values. The OS should disable
240  *        the SCI interrupt level when the handler is removed, so no more
241  *        events will come in.
242  *
243  *****************************************************************************/

245 ACPI_STATUS
246 AcpiEvRemoveAllSciHandlers (
247     void)
248 {
249     ACPI_SCI_HANDLER_INFO   *SciHandler;
250     ACPI_CPU_FLAGS          Flags;
251     ACPI_STATUS             Status;


254     ACPI_FUNCTION_TRACE (EvRemoveAllSciHandlers);


257     /* Just let the OS remove the handler and disable the level */
```

```
259     Status = AcpiOsRemoveInterruptHandler ((UINT32) AcpiGbl_FADT.SciInterrupt,
260                 AcpiEvSciXruptHandler);
261
262     if (!AcpiGbl_SciHandlerList)
263     {
264         return (Status);
265     }
266
267     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
268
269     /* Free all host-installed SCI handlers */
270
271     while (AcpiGbl_SciHandlerList)
272     {
273         SciHandler = AcpiGbl_SciHandlerList;
274         AcpiGbl_SciHandlerList = SciHandler->Next;
275         ACPI_FREE (SciHandler);
276     }
277
278     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
279     return_ACPI_STATUS (Status);
280 }
281
282 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   30979 Thu Dec 26 13:48:59 2013
new/usr/src/common/acpica/components/events/evxface.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evxface - External interfaces for ACPI events
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EVXFACE_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acevents.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evxface")


/******************************************************************************
 *
```



```
 * FUNCTION:    AcpiInstallNotifyHandler
 *
 * PARAMETERS:  Device          - The device for which notifies will be handled
 *              HandlerType     - The type of handler:
 *                                  ACPI_SYSTEM_NOTIFY: System Handler (00-7F)
 *                                  ACPI_DEVICE_NOTIFY: Device Handler (80-FF)
 *                                  ACPI_ALL_NOTIFY:    Both System and Device
 *              Handler         - Address of the handler
 *              Context         - Value passed to the handler on each GPE
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Install a handler for notifications on an ACPI Device,
 *              ThermalZone, or Processor object.
 *
 * NOTES:       The Root namespace object may have only one handler for each
 *              type of notify (System/Device). Device/Thermal/Processor objects
 *              may have one device notify handler, and multiple system notify
 *              handlers.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiInstallNotifyHandler (
    ACPI_HANDLE             Device,
    UINT32                  HandlerType,
    ACPI_NOTIFY_HANDLER     Handler,
    void                    *Context)
{
    ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Device);
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     *HandlerObj;
    ACPI_STATUS             Status;
    UINT32                  i;


    ACPI_FUNCTION_TRACE (AcpiInstallNotifyHandler);


    /* Parameter validation */

    if ((!Device) || (!Handler) || (!HandlerType) ||
        (HandlerType > ACPI_MAX_NOTIFY_HANDLER_TYPE))
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * Root Object:
     * Registering a notify handler on the root object indicates that the
     * caller wishes to receive notifications for all objects. Note that
     * only one global handler can be registered per notify type.
     * Ensure that a handler is not already installed.
     */
    if (Device == ACPI_ROOT_OBJECT)
    {
        for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
        {
            if (HandlerType & (i+1))
            {
```

```
126                 if (AcpiGbl_GlobalNotify[i].Handler)
127                 {
128                     Status = AE_ALREADY_EXISTS;
129                     goto UnlockAndExit;
130                 }
131
132                 AcpiGbl_GlobalNotify[i].Handler = Handler;
133                 AcpiGbl_GlobalNotify[i].Context = Context;
134             }
135         }
136
137         goto UnlockAndExit; /* Global notify handler installed, all done */
138     }
139
140     /*
141      * All Other Objects:
142      * Caller will only receive notifications specific to the target
143      * object. Note that only certain object types are allowed to
144      * receive notifications.
145      */
146
147     /* Are Notifies allowed on this object? */
148
149     if (!AcpiEvIsNotifyObject (Node))
150     {
151         Status = AE_TYPE;
152         goto UnlockAndExit;
153     }
154
155     /* Check for an existing internal object, might not exist */
156
157     ObjDesc = AcpiNsGetAttachedObject (Node);
158     if (!ObjDesc)
159     {
160         /* Create a new object */
161
162         ObjDesc = AcpiUtCreateInternalObject (Node->Type);
163         if (!ObjDesc)
164         {
165             Status = AE_NO_MEMORY;
166             goto UnlockAndExit;
167         }
168
169         /* Attach new object to the Node, remove local reference */
170
171         Status = AcpiNsAttachObject (Device, ObjDesc, Node->Type);
172         AcpiUtRemoveReference (ObjDesc);
173         if (ACPI_FAILURE (Status))
174         {
175             goto UnlockAndExit;
176         }
177     }
178
179     /* Ensure that the handler is not already installed in the lists */
180
181     for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
182     {
183         if (HandlerType & (i+1))
184         {
185             HandlerObj = ObjDesc->CommonNotify.NotifyList[i];
186             while (HandlerObj)
187             {
188                 if (HandlerObj->Notify.Handler == Handler)
189                 {
190                     Status = AE_ALREADY_EXISTS;
191                     goto UnlockAndExit;
```



```
192                 }
193
194                 HandlerObj = HandlerObj->Notify.Next[i];
195             }
196         }
197     }
198
199     /* Create and populate a new notify handler object */
200
201     HandlerObj = AcpiUtCreateInternalObject (ACPI_TYPE_LOCAL_NOTIFY);
202     if (!HandlerObj)
203     {
204         Status = AE_NO_MEMORY;
205         goto UnlockAndExit;
206     }
207
208     HandlerObj->Notify.Node = Node;
209     HandlerObj->Notify.HandlerType = HandlerType;
210     HandlerObj->Notify.Handler = Handler;
211     HandlerObj->Notify.Context = Context;
212
213     /* Install the handler at the list head(s) */
214
215     for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
216     {
217         if (HandlerType & (i+1))
218         {
219             HandlerObj->Notify.Next[i] =
220                 ObjDesc->CommonNotify.NotifyList[i];
221
222             ObjDesc->CommonNotify.NotifyList[i] = HandlerObj;
223         }
224     }
225
226     /* Add an extra reference if handler was installed in both lists */
227
228     if (HandlerType == ACPI_ALL_NOTIFY)
229     {
230         AcpiUtAddReference (HandlerObj);
231     }
232
233
234 UnlockAndExit:
235     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
236     return_ACPI_STATUS (Status);
237 }
238
239 ACPI_EXPORT_SYMBOL (AcpiInstallNotifyHandler)
240
241
242 /*******************************************************************************
243  *
244  * FUNCTION:    AcpiRemoveNotifyHandler
245  *
246  * PARAMETERS:  Device          - The device for which the handler is installed
247  *              HandlerType     - The type of handler:
248  *                                  ACPI_SYSTEM_NOTIFY: System Handler (00-7F)
249  *                                  ACPI_DEVICE_NOTIFY: Device Handler (80-FF)
250  *                                  ACPI_ALL_NOTIFY:    Both System and Device
251  *              Handler         - Address of the handler
252  *
253  * RETURN:      Status
254  *
255  * DESCRIPTION: Remove a handler for notifies on an ACPI device
256  *
257  ******************************************************************************/
```

```
ACPI_STATUS
AcpiRemoveNotifyHandler (
    ACPI_HANDLE             Device,
    UINT32                  HandlerType,
    ACPI_NOTIFY_HANDLER     Handler)
{
    ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Device);
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     *HandlerObj;
    ACPI_OPERAND_OBJECT     *PreviousHandlerObj;
    ACPI_STATUS             Status;
    UINT32                  i;


    ACPI_FUNCTION_TRACE (AcpiRemoveNotifyHandler);


    /* Parameter validation */

    if ((!Device) || (!Handler) || (!HandlerType) ||
        (HandlerType > ACPI_MAX_NOTIFY_HANDLER_TYPE))
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* Make sure all deferred notify tasks are completed */

    AcpiOsWaitEventsComplete ();

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Root Object. Global handlers are removed here */

    if (Device == ACPI_ROOT_OBJECT)
    {
        for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
        {
            if (HandlerType & (i+1))
            {
                if (!AcpiGbl_GlobalNotify[i].Handler ||
                    (AcpiGbl_GlobalNotify[i].Handler != Handler))
                {
                    Status = AE_NOT_EXIST;
                    goto UnlockAndExit;
                }

                ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
                    "Removing global notify handler\n"));

                AcpiGbl_GlobalNotify[i].Handler = NULL;
                AcpiGbl_GlobalNotify[i].Context = NULL;
            }
        }

        goto UnlockAndExit;
    }

    /* All other objects: Are Notifies allowed on this object? */

    if (!AcpiEvIsNotifyObject (Node))
    {
```



```
        Status = AE_TYPE;
        goto UnlockAndExit;
    }

    /* Must have an existing internal object */

    ObjDesc = AcpiNsGetAttachedObject (Node);
    if (!ObjDesc)
    {
        Status = AE_NOT_EXIST;
        goto UnlockAndExit;
    }

    /* Internal object exists. Find the handler and remove it */

    for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
    {
        if (HandlerType & (i+1))
        {
            HandlerObj = ObjDesc->CommonNotify.NotifyList[i];
            PreviousHandlerObj = NULL;

            /* Attempt to find the handler in the handler list */

            while (HandlerObj &&
                  (HandlerObj->Notify.Handler != Handler))
            {
                PreviousHandlerObj = HandlerObj;
                HandlerObj = HandlerObj->Notify.Next[i];
            }

            if (!HandlerObj)
            {
                Status = AE_NOT_EXIST;
                goto UnlockAndExit;
            }

            /* Remove the handler object from the list */

            if (PreviousHandlerObj) /* Handler is not at the list head */
            {
                PreviousHandlerObj->Notify.Next[i] =
                    HandlerObj->Notify.Next[i];
            }
            else /* Handler is at the list head */
            {
                ObjDesc->CommonNotify.NotifyList[i] =
                    HandlerObj->Notify.Next[i];
            }

            AcpiUtRemoveReference (HandlerObj);
        }
    }


UnlockAndExit:
    (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiRemoveNotifyHandler)


/******************************************************************************
 *
 * FUNCTION:    AcpiInstallExceptionHandler
```

```
 *
 * PARAMETERS:  Handler         - Pointer to the handler function for the
 *                                event
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Saves the pointer to the handler function
 *
 ******************************************************************************/

ACPI_STATUS
AcpiInstallExceptionHandler (
    ACPI_EXCEPTION_HANDLER  Handler)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiInstallExceptionHandler);


    Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Don't allow two handlers. */

    if (AcpiGbl_ExceptionHandler)
    {
        Status = AE_ALREADY_EXISTS;
        goto Cleanup;
    }

    /* Install the handler */

    AcpiGbl_ExceptionHandler = Handler;

Cleanup:
    (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiInstallExceptionHandler)


#if (!ACPI_REDUCED_HARDWARE)
/******************************************************************************
 *
 * FUNCTION:    AcpiInstallSciHandler
 *
 * PARAMETERS:  Address             - Address of the handler
 *              Context             - Value passed to the handler on each SCI
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Install a handler for a System Control Interrupt.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiInstallSciHandler (
    ACPI_SCI_HANDLER        Address,
    void                    *Context)
{
    ACPI_SCI_HANDLER_INFO   *NewSciHandler;
```



```
    ACPI_SCI_HANDLER_INFO   *SciHandler;
    ACPI_CPU_FLAGS          Flags;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiInstallSciHandler);


    if (!Address)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* Allocate and init a handler object */

    NewSciHandler = ACPI_ALLOCATE (sizeof (ACPI_SCI_HANDLER_INFO));
    if (!NewSciHandler)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    NewSciHandler->Address = Address;
    NewSciHandler->Context = Context;

    Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
    if (ACPI_FAILURE (Status))
    {
        goto Exit;
    }

    /* Lock list during installation */

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
    SciHandler = AcpiGbl_SciHandlerList;

    /* Ensure handler does not already exist */

    while (SciHandler)
    {
        if (Address == SciHandler->Address)
        {
            Status = AE_ALREADY_EXISTS;
            goto UnlockAndExit;
        }

        SciHandler = SciHandler->Next;
    }

    /* Install the new handler into the global list (at head) */

    NewSciHandler->Next = AcpiGbl_SciHandlerList;
    AcpiGbl_SciHandlerList = NewSciHandler;


UnlockAndExit:

    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);

Exit:
    if (ACPI_FAILURE (Status))
    {
        ACPI_FREE (NewSciHandler);
    }
    return_ACPI_STATUS (Status);
}
```

```
524 /*******************************************************************************
525  *
526  * FUNCTION:    AcpiRemoveSciHandler
527  *
528  * PARAMETERS:  Address             - Address of the handler
529  *
530  * RETURN:      Status
531  *
532  * DESCRIPTION: Remove a handler for a System Control Interrupt.
533  *
534  ******************************************************************************/

536 ACPI_STATUS
537 AcpiRemoveSciHandler (
538     ACPI_SCI_HANDLER        Address)
539 {
540     ACPI_SCI_HANDLER_INFO   *PrevSciHandler;
541     ACPI_SCI_HANDLER_INFO   *NextSciHandler;
542     ACPI_CPU_FLAGS          Flags;
543     ACPI_STATUS             Status;


546     ACPI_FUNCTION_TRACE (AcpiRemoveSciHandler);


549     if (!Address)
550     {
551         return_ACPI_STATUS (AE_BAD_PARAMETER);
552     }

554     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
555     if (ACPI_FAILURE (Status))
556     {
557         return_ACPI_STATUS (Status);
558     }

560     /* Remove the SCI handler with lock */

562     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

564     PrevSciHandler = NULL;
565     NextSciHandler = AcpiGbl_SciHandlerList;
566     while (NextSciHandler)
567     {
568         if (NextSciHandler->Address == Address)
569         {
570             /* Unlink and free the SCI handler info block */

572             if (PrevSciHandler)
573             {
574                 PrevSciHandler->Next = NextSciHandler->Next;
575             }
576             else
577             {
578                 AcpiGbl_SciHandlerList = NextSciHandler->Next;
579             }

581             AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
582             ACPI_FREE (NextSciHandler);
583             goto UnlockAndExit;
584         }

586         PrevSciHandler = NextSciHandler;
587         NextSciHandler = NextSciHandler->Next;
```



```
588     }

590     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
591     Status = AE_NOT_EXIST;


594 UnlockAndExit:
595     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
596     return_ACPI_STATUS (Status);
597 }


600 /*******************************************************************************
601  *
602  * FUNCTION:    AcpiInstallGlobalEventHandler
603  *
604  * PARAMETERS:  Handler         - Pointer to the global event handler function
605  *              Context         - Value passed to the handler on each event
606  *
607  * RETURN:      Status
608  *
609  * DESCRIPTION: Saves the pointer to the handler function. The global handler
610  *              is invoked upon each incoming GPE and Fixed Event. It is
611  *              invoked at interrupt level at the time of the event dispatch.
612  *              Can be used to update event counters, etc.
613  *
614  ******************************************************************************/

616 ACPI_STATUS
617 AcpiInstallGlobalEventHandler (
618     ACPI_GBL_EVENT_HANDLER  Handler,
619     void                    *Context)
620 {
621     ACPI_STATUS             Status;


624     ACPI_FUNCTION_TRACE (AcpiInstallGlobalEventHandler);


627     /* Parameter validation */

629     if (!Handler)
630     {
631         return_ACPI_STATUS (AE_BAD_PARAMETER);
632     }

634     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
635     if (ACPI_FAILURE (Status))
636     {
637         return_ACPI_STATUS (Status);
638     }

640     /* Don't allow two handlers. */

642     if (AcpiGbl_GlobalEventHandler)
643     {
644         Status = AE_ALREADY_EXISTS;
645         goto Cleanup;
646     }

648     AcpiGbl_GlobalEventHandler = Handler;
649     AcpiGbl_GlobalEventHandlerContext = Context;


652 Cleanup:
653     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
```

```
654     return_ACPI_STATUS (Status);
655 }

657 ACPI_EXPORT_SYMBOL (AcpiInstallGlobalEventHandler)


660 /*******************************************************************************
661  *
662  * FUNCTION:    AcpiInstallFixedEventHandler
663  *
664  * PARAMETERS:  Event           - Event type to enable.
665  *              Handler         - Pointer to the handler function for the
666  *                                event
667  *              Context         - Value passed to the handler on each GPE
668  *
669  * RETURN:      Status
670  *
671  * DESCRIPTION: Saves the pointer to the handler function and then enables the
672  *              event.
673  *
674  ******************************************************************************/

676 ACPI_STATUS
677 AcpiInstallFixedEventHandler (
678     UINT32                  Event,
679     ACPI_EVENT_HANDLER      Handler,
680     void                    *Context)
681 {
682     ACPI_STATUS             Status;


685     ACPI_FUNCTION_TRACE (AcpiInstallFixedEventHandler);


688     /* Parameter validation */

690     if (Event > ACPI_EVENT_MAX)
691     {
692         return_ACPI_STATUS (AE_BAD_PARAMETER);
693     }

695     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
696     if (ACPI_FAILURE (Status))
697     {
698         return_ACPI_STATUS (Status);
699     }

701     /* Do not allow multiple handlers */

703     if (AcpiGbl_FixedEventHandlers[Event].Handler)
704     {
705         Status = AE_ALREADY_EXISTS;
706         goto Cleanup;
707     }

709     /* Install the handler before enabling the event */

711     AcpiGbl_FixedEventHandlers[Event].Handler = Handler;
712     AcpiGbl_FixedEventHandlers[Event].Context = Context;

714     Status = AcpiEnableEvent (Event, 0);
715     if (ACPI_FAILURE (Status))
716     {
717         ACPI_WARNING ((AE_INFO,
718             "Could not enable fixed event - %s (%u)",
719             AcpiUtGetEventName (Event), Event));
```



```
721         /* Remove the handler */

723         AcpiGbl_FixedEventHandlers[Event].Handler = NULL;
724         AcpiGbl_FixedEventHandlers[Event].Context = NULL;
725     }
726     else
727     {
728         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
729             "Enabled fixed event %s (%X), Handler=%p\n",
730             AcpiUtGetEventName (Event), Event, Handler));
731     }


734 Cleanup:
735     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
736     return_ACPI_STATUS (Status);
737 }

739 ACPI_EXPORT_SYMBOL (AcpiInstallFixedEventHandler)


742 /*******************************************************************************
743  *
744  * FUNCTION:    AcpiRemoveFixedEventHandler
745  *
746  * PARAMETERS:  Event           - Event type to disable.
747  *              Handler         - Address of the handler
748  *
749  * RETURN:      Status
750  *
751  * DESCRIPTION: Disables the event and unregisters the event handler.
752  *
753  ******************************************************************************/

755 ACPI_STATUS
756 AcpiRemoveFixedEventHandler (
757     UINT32                  Event,
758     ACPI_EVENT_HANDLER      Handler)
759 {
760     ACPI_STATUS             Status = AE_OK;


763     ACPI_FUNCTION_TRACE (AcpiRemoveFixedEventHandler);


766     /* Parameter validation */

768     if (Event > ACPI_EVENT_MAX)
769     {
770         return_ACPI_STATUS (AE_BAD_PARAMETER);
771     }

773     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
774     if (ACPI_FAILURE (Status))
775     {
776         return_ACPI_STATUS (Status);
777     }

779     /* Disable the event before removing the handler */

781     Status = AcpiDisableEvent (Event, 0);

783     /* Always Remove the handler */

785     AcpiGbl_FixedEventHandlers[Event].Handler = NULL;
```

```
786     AcpiGbl_FixedEventHandlers[Event].Context = NULL;

788     if (ACPI_FAILURE (Status))
789     {
790         ACPI_WARNING ((AE_INFO,
791             "Could not disable fixed event - %s (%u)",
792             AcpiUtGetEventName (Event), Event));
793     }
794     else
795     {
796         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
797             "Disabled fixed event - %s (%X)\n",
798             AcpiUtGetEventName (Event), Event));
799     }

801     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
802     return_ACPI_STATUS (Status);
803 }

805 ACPI_EXPORT_SYMBOL (AcpiRemoveFixedEventHandler)


808 /******************************************************************************
809  *
810  * FUNCTION:    AcpiInstallGpeHandler
811  *
812  * PARAMETERS:  GpeDevice       - Namespace node for the GPE (NULL for FADT
813  *                                defined GPEs)
814  *              GpeNumber       - The GPE number within the GPE block
815  *              Type            - Whether this GPE should be treated as an
816  *                                edge- or level-triggered interrupt.
817  *              Address         - Address of the handler
818  *              Context         - Value passed to the handler on each GPE
819  *
820  * RETURN:      Status
821  *
822  * DESCRIPTION: Install a handler for a General Purpose Event.
823  *
824  *****************************************************************************/

826 ACPI_STATUS
827 AcpiInstallGpeHandler (
828     ACPI_HANDLE             GpeDevice,
829     UINT32                  GpeNumber,
830     UINT32                  Type,
831     ACPI_GPE_HANDLER        Address,
832     void                    *Context)
833 {
834     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
835     ACPI_GPE_HANDLER_INFO   *Handler;
836     ACPI_STATUS             Status;
837     ACPI_CPU_FLAGS          Flags;


840     ACPI_FUNCTION_TRACE (AcpiInstallGpeHandler);


843     /* Parameter validation */

845     if ((!Address) || (Type & ~ACPI_GPE_XRUPT_TYPE_MASK))
846     {
847         return_ACPI_STATUS (AE_BAD_PARAMETER);
848     }

850     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
851     if (ACPI_FAILURE (Status))
```



```
852     {
853         return_ACPI_STATUS (Status);
854     }

856     /* Allocate and init handler object (before lock) */

858     Handler = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_HANDLER_INFO));
859     if (!Handler)
860     {
861         Status = AE_NO_MEMORY;
862         goto UnlockAndExit;
863     }

865     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

867     /* Ensure that we have a valid GPE number */

869     GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
870     if (!GpeEventInfo)
871     {
872         Status = AE_BAD_PARAMETER;
873         goto FreeAndExit;
874     }

876     /* Make sure that there isn't a handler there already */

878     if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
879             ACPI_GPE_DISPATCH_HANDLER)
880     {
881         Status = AE_ALREADY_EXISTS;
882         goto FreeAndExit;
883     }

885     Handler->Address = Address;
886     Handler->Context = Context;
887     Handler->MethodNode = GpeEventInfo->Dispatch.MethodNode;
888     Handler->OriginalFlags = (UINT8) (GpeEventInfo->Flags &
889         (ACPI_GPE_XRUPT_TYPE_MASK | ACPI_GPE_DISPATCH_MASK));

891     /*
892      * If the GPE is associated with a method, it may have been enabled
893      * automatically during initialization, in which case it has to be
894      * disabled now to avoid spurious execution of the handler.
895      */
896     if (((Handler->OriginalFlags & ACPI_GPE_DISPATCH_METHOD) ||
897          (Handler->OriginalFlags & ACPI_GPE_DISPATCH_NOTIFY)) &&
898         GpeEventInfo->RuntimeCount)
899     {
900         Handler->OriginallyEnabled = TRUE;
901         (void) AcpiEvRemoveGpeReference (GpeEventInfo);

903         /* Sanity check of original type against new type */

905         if (Type != (UINT32) (GpeEventInfo->Flags & ACPI_GPE_XRUPT_TYPE_MASK))
906         {
907             ACPI_WARNING ((AE_INFO, "GPE type mismatch (level/edge)"));
908         }
909     }

911     /* Install the handler */

913     GpeEventInfo->Dispatch.Handler = Handler;

915     /* Setup up dispatch flags to indicate handler (vs. method/notify) */

917     GpeEventInfo->Flags &= ~(ACPI_GPE_XRUPT_TYPE_MASK | ACPI_GPE_DISPATCH_MASK);
```

```
    GpeEventInfo->Flags |= (UINT8) (Type | ACPI_GPE_DISPATCH_HANDLER);

    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);


UnlockAndExit:
    (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
    return_ACPI_STATUS (Status);

FreeAndExit:
    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    ACPI_FREE (Handler);
    goto UnlockAndExit;
}

ACPI_EXPORT_SYMBOL (AcpiInstallGpeHandler)


/******************************************************************************
 *
 * FUNCTION:    AcpiRemoveGpeHandler
 *
 * PARAMETERS:  GpeDevice       - Namespace node for the GPE (NULL for FADT
 *                                defined GPEs)
 *              GpeNumber       - The event to remove a handler
 *              Address         - Address of the handler
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Remove a handler for a General Purpose AcpiEvent.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiRemoveGpeHandler (
    ACPI_HANDLE             GpeDevice,
    UINT32                  GpeNumber,
    ACPI_GPE_HANDLER        Address)
{
    ACPI_GPE_EVENT_INFO     *GpeEventInfo;
    ACPI_GPE_HANDLER_INFO   *Handler;
    ACPI_STATUS             Status;
    ACPI_CPU_FLAGS          Flags;


    ACPI_FUNCTION_TRACE (AcpiRemoveGpeHandler);


    /* Parameter validation */

    if (!Address)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* Make sure all deferred GPE tasks are completed */

    AcpiOsWaitEventsComplete ();

    Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
```



```
    /* Ensure that we have a valid GPE number */

    GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
    if (!GpeEventInfo)
    {
        Status = AE_BAD_PARAMETER;
        goto UnlockAndExit;
    }

    /* Make sure that a handler is indeed installed */

    if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) !=
            ACPI_GPE_DISPATCH_HANDLER)
    {
        Status = AE_NOT_EXIST;
        goto UnlockAndExit;
    }

    /* Make sure that the installed handler is the same */

    if (GpeEventInfo->Dispatch.Handler->Address != Address)
    {
        Status = AE_BAD_PARAMETER;
        goto UnlockAndExit;
    }

    /* Remove the handler */

    Handler = GpeEventInfo->Dispatch.Handler;

    /* Restore Method node (if any), set dispatch flags */

    GpeEventInfo->Dispatch.MethodNode = Handler->MethodNode;
    GpeEventInfo->Flags &=
        ~(ACPI_GPE_XRUPT_TYPE_MASK | ACPI_GPE_DISPATCH_MASK);
    GpeEventInfo->Flags |= Handler->OriginalFlags;

    /*
     * If the GPE was previously associated with a method and it was
     * enabled, it should be enabled at this point to restore the
     * post-initialization configuration.
     */
    if ((Handler->OriginalFlags & ACPI_GPE_DISPATCH_METHOD) &&
        Handler->OriginallyEnabled)
    {
        (void) AcpiEvAddGpeReference (GpeEventInfo);
    }

    /* Now we can free the handler object */

    ACPI_FREE (Handler);


UnlockAndExit:
    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
    (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiRemoveGpeHandler)


/******************************************************************************
 *
 * FUNCTION:    AcpiAcquireGlobalLock
```

```
 *
 * PARAMETERS:  Timeout         - How long the caller is willing to wait
 *              Handle          - Where the handle to the lock is returned
 *                                (if acquired)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Acquire the ACPI Global Lock
 *
 * Note: Allows callers with the same thread ID to acquire the global lock
 * multiple times. In other words, externally, the behavior of the global lock
 * is identical to an AML mutex. On the first acquire, a new handle is
 * returned. On any subsequent calls to acquire by the same thread, the same
 * handle is returned.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiAcquireGlobalLock (
    UINT16                  Timeout,
    UINT32                  *Handle)
{
    ACPI_STATUS             Status;


    if (!Handle)
    {
        return (AE_BAD_PARAMETER);
    }

    /* Must lock interpreter to prevent race conditions */

    AcpiExEnterInterpreter ();

    Status = AcpiExAcquireMutexObject (Timeout,
                AcpiGbl_GlobalLockMutex, AcpiOsGetThreadId ());

    if (ACPI_SUCCESS (Status))
    {
        /* Return the global lock handle (updated in AcpiEvAcquireGlobalLock) */

        *Handle = AcpiGbl_GlobalLockHandle;
    }

    AcpiExExitInterpreter ();
    return (Status);
}

ACPI_EXPORT_SYMBOL (AcpiAcquireGlobalLock)


/*****************************************************************************
 *
 * FUNCTION:    AcpiReleaseGlobalLock
 *
 * PARAMETERS:  Handle      - Returned from AcpiAcquireGlobalLock
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Release the ACPI Global Lock. The handle must be valid.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiReleaseGlobalLock (
    UINT32                  Handle)
```



```
{
    ACPI_STATUS             Status;


    if (!Handle || (Handle != AcpiGbl_GlobalLockHandle))
    {
        return (AE_NOT_ACQUIRED);
    }

    Status = AcpiExReleaseMutexObject (AcpiGbl_GlobalLockMutex);
    return (Status);
}

ACPI_EXPORT_SYMBOL (AcpiReleaseGlobalLock)

#endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   10127 Thu Dec 26 13:48:59 2013
new/usr/src/common/acpica/components/events/evxfevnt.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evxfevnt - External Interfaces, ACPI event disable/enable
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EVXFEVNT_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evxfevnt")


#if (!ACPI_REDUCED_HARDWARE) /* Entire module */
/******************************************************************************
 *
```



```
 * FUNCTION:    AcpiEnable
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Transfers the system into ACPI mode.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiEnable (
    void)
{
    ACPI_STATUS             Status = AE_OK;


    ACPI_FUNCTION_TRACE (AcpiEnable);


    /* ACPI tables must be present */

    if (!AcpiTbTablesLoaded ())
    {
        return_ACPI_STATUS (AE_NO_ACPI_TABLES);
    }

    /* If the Hardware Reduced flag is set, machine is always in acpi mode */

    if (AcpiGbl_ReducedHardware)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /* Check current mode */

    if (AcpiHwGetMode() == ACPI_SYS_MODE_ACPI)
    {
        ACPI_DEBUG_PRINT ((ACPI_DB_INIT, "System is already in ACPI mode\n"));
    }
    else
    {
        /* Transition to ACPI mode */

        Status = AcpiHwSetMode (ACPI_SYS_MODE_ACPI);
        if (ACPI_FAILURE (Status))
        {
            ACPI_ERROR ((AE_INFO, "Could not transition to ACPI mode"));
            return_ACPI_STATUS (Status);
        }

        ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
            "Transition to ACPI mode successful\n"));
    }

    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiEnable)


/******************************************************************************
 *
 * FUNCTION:    AcpiDisable
 *
 * PARAMETERS:  None
```

```
 125  *
 126  * RETURN:      Status
 127  *
 128  * DESCRIPTION: Transfers the system into LEGACY (non-ACPI) mode.
 129  *
 130  ******************************************************************************/

 132 ACPI_STATUS
 133 AcpiDisable (
 134     void)
 135 {
 136     ACPI_STATUS             Status = AE_OK;


 139     ACPI_FUNCTION_TRACE (AcpiDisable);


 142     /* If the Hardware Reduced flag is set, machine is always in acpi mode */

 144     if (AcpiGbl_ReducedHardware)
 145     {
 146         return_ACPI_STATUS (AE_OK);
 147     }

 149     if (AcpiHwGetMode() == ACPI_SYS_MODE_LEGACY)
 150     {
 151         ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
 152             "System is already in legacy (non-ACPI) mode\n"));
 153     }
 154     else
 155     {
 156         /* Transition to LEGACY mode */

 158         Status = AcpiHwSetMode (ACPI_SYS_MODE_LEGACY);

 160         if (ACPI_FAILURE (Status))
 161         {
 162             ACPI_ERROR ((AE_INFO,
 163                 "Could not exit ACPI mode to legacy mode"));
 164             return_ACPI_STATUS (Status);
 165         }

 167         ACPI_DEBUG_PRINT ((ACPI_DB_INIT, "ACPI mode disabled\n"));
 168     }

 170     return_ACPI_STATUS (Status);
 171 }
```

*unchanged_portion_omitted_*

```
 394 ACPI_EXPORT_SYMBOL (AcpiGetEventStatus)

 396 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   26082 Thu Dec 26 13:49:00 2013
new/usr/src/common/acpica/components/events/evxfgpe.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: evxfgpe - External Interfaces for General Purpose Events (GPEs)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __EVXFGPE_C__
  46 #define EXPORT_ACPI_INTERFACES

  48 #include "acpi.h"
  49 #include "accommon.h"
  50 #include "acevents.h"
  51 #include "acnamesp.h"

  53 #define _COMPONENT          ACPI_EVENTS
  54         ACPI_MODULE_NAME    ("evxfgpe")


  57 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
  58 /******************************************************************************
```



```
  59  *
  60  * FUNCTION:    AcpiUpdateAllGpes
  61  *
  62  * PARAMETERS:  None
  63  *
  64  * RETURN:      Status
  65  *
  66  * DESCRIPTION: Complete GPE initialization and enable all GPEs that have
  67  *              associated _Lxx or _Exx methods and are not pointed to by any
  68  *              device _PRW methods (this indicates that these GPEs are
  69  *              generally intended for system or device wakeup. Such GPEs
  70  *              have to be enabled directly when the devices whose _PRW
  71  *              methods point to them are set up for wakeup signaling.)
  72  *
  73  * NOTE: Should be called after any GPEs are added to the system. Primarily,
  74  * after the system _PRW methods have been run, but also after a GPE Block
  75  * Device has been added or if any new GPE methods have been added via a
  76  * dynamic table load.
  77  *
  78  *****************************************************************************/

  80 ACPI_STATUS
  81 AcpiUpdateAllGpes (
  82     void)
  83 {
  84     ACPI_STATUS             Status;


  87     ACPI_FUNCTION_TRACE (AcpiUpdateAllGpes);
  85     ACPI_FUNCTION_TRACE (AcpiUpdateGpes);


  90     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
  91     if (ACPI_FAILURE (Status))
  92     {
  93         return_ACPI_STATUS (Status);
  94     }

  96     if (AcpiGbl_AllGpesInitialized)
  97     {
  98         goto UnlockAndExit;
  99     }

 101     Status = AcpiEvWalkGpeList (AcpiEvInitializeGpeBlock, NULL);
 102     if (ACPI_SUCCESS (Status))
 103     {
 104         AcpiGbl_AllGpesInitialized = TRUE;
 105     }

 107 UnlockAndExit:
 108     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
 109     return_ACPI_STATUS (Status);
 110 }
______unchanged_portion_omitted_

 201 ACPI_EXPORT_SYMBOL (AcpiDisableGpe)


 204 /******************************************************************************
 205  *
 206  * FUNCTION:    AcpiSetGpe
 207  *
 208  * PARAMETERS:  GpeDevice           - Parent GPE Device. NULL for GPE0/GPE1
 209  *              GpeNumber           - GPE level within the GPE block
 210  *              Action              - ACPI_GPE_ENABLE or ACPI_GPE_DISABLE
 211  *
```

```
 212  * RETURN:      Status
 213  *
 214  * DESCRIPTION: Enable or disable an individual GPE. This function bypasses
 215  *              the reference count mechanism used in the AcpiEnableGpe and
 216  *              AcpiDisableGpe interfaces -- and should be used with care.
 217  *
 218  * Note: Typically used to disable a runtime GPE for short period of time,
 219  * then re-enable it, without disturbing the existing reference counts. This
 220  * is useful, for example, in the Embedded Controller (EC) driver.
 221  *
 222  ******************************************************************************/

 224 ACPI_STATUS
 225 AcpiSetGpe (
 226     ACPI_HANDLE             GpeDevice,
 227     UINT32                  GpeNumber,
 228     UINT8                   Action)
 229 {
 230     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
 231     ACPI_STATUS             Status;
 232     ACPI_CPU_FLAGS          Flags;


 235     ACPI_FUNCTION_TRACE (AcpiSetGpe);


 238     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

 240     /* Ensure that we have a valid GPE number */

 242     GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
 243     if (!GpeEventInfo)
 244     {
 245         Status = AE_BAD_PARAMETER;
 246         goto UnlockAndExit;
 247     }

 249     /* Perform the action */

 251     switch (Action)
 252     {
 253     case ACPI_GPE_ENABLE:

 255         Status = AcpiEvEnableGpe (GpeEventInfo);
 256         break;

 258     case ACPI_GPE_DISABLE:

 260         Status = AcpiHwLowSetGpe (GpeEventInfo, ACPI_GPE_DISABLE);
 261         break;

 263     default:

 265         Status = AE_BAD_PARAMETER;
 266         break;
 267     }

 269 UnlockAndExit:
 270     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
 271     return_ACPI_STATUS (Status);
 272 }

 274 ACPI_EXPORT_SYMBOL (AcpiSetGpe)


 277 /*******************************************************************************
```



```
 278  *
 279  * FUNCTION:    AcpiSetupGpeForWake
 280  *
 281  * PARAMETERS:  WakeDevice          - Device associated with the GPE (via _PRW)
 282  *              GpeDevice           - Parent GPE Device. NULL for GPE0/GPE1
 283  *              GpeNumber           - GPE level within the GPE block
 284  *
 285  * RETURN:      Status
 286  *
 287  * DESCRIPTION: Mark a GPE as having the ability to wake the system. This
 288  *              interface is intended to be used as the host executes the
 289  *              _PRW methods (Power Resources for Wake) in the system tables.
 290  *              Each _PRW appears under a Device Object (The WakeDevice), and
 291  *              contains the info for the wake GPE associated with the
 292  *              WakeDevice.
 293  *
 294  ******************************************************************************/

 296 ACPI_STATUS
 297 AcpiSetupGpeForWake (
 298     ACPI_HANDLE             WakeDevice,
 299     ACPI_HANDLE             GpeDevice,
 300     UINT32                  GpeNumber)
 301 {
 302     ACPI_STATUS             Status;
 297     ACPI_STATUS             Status = AE_BAD_PARAMETER;
 303     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
 304     ACPI_NAMESPACE_NODE     *DeviceNode;
 305     ACPI_GPE_NOTIFY_INFO    *Notify;
 306     ACPI_GPE_NOTIFY_INFO    *NewNotify;
 307     ACPI_CPU_FLAGS          Flags;


 310     ACPI_FUNCTION_TRACE (AcpiSetupGpeForWake);


 313     /* Parameter Validation */

 315     if (!WakeDevice)
 316     {
 317         /*
 318          * By forcing WakeDevice to be valid, we automatically enable the
 319          * implicit notify feature on all hosts.
 320          */
 321         return_ACPI_STATUS (AE_BAD_PARAMETER);
 322     }

 324     /* Handle root object case */

 326     if (WakeDevice == ACPI_ROOT_OBJECT)
 327     {
 328         DeviceNode = AcpiGbl_RootNode;
 329     }
 330     else
 331     {
 332         DeviceNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, WakeDevice);
 333     }

 335     /* Validate WakeDevice is of type Device */

 337     if (DeviceNode->Type != ACPI_TYPE_DEVICE)
 338     {
 339         return_ACPI_STATUS (AE_BAD_PARAMETER);
 340     }

 342     /*
```

```
     * Allocate a new notify object up front, in case it is needed.
     * Memory allocation while holding a spinlock is a big no-no
     * on some hosts.
     */
    NewNotify = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_NOTIFY_INFO));
    if (!NewNotify)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

    /* Ensure that we have a valid GPE number */

    GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
    if (!GpeEventInfo)
    if (GpeEventInfo)
    {
        Status = AE_BAD_PARAMETER;
        goto UnlockAndExit;
    }

    /*
     * If there is no method or handler for this GPE, then the
     * WakeDevice will be notified whenever this GPE fires. This is
     * known as an "implicit notify". Note: The GPE is assumed to be
      * WakeDevice will be notified whenever this GPE fires (aka
      * "implicit notify") Note: The GPE is assumed to be
     * level-triggered (for windows compatibility).
     */
    if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
            ACPI_GPE_DISPATCH_NONE)
    {
        /*
         * This is the first device for implicit notify on this GPE.
         * Just set the flags here, and enter the NOTIFY block below.
         */
        GpeEventInfo->Flags =
            (ACPI_GPE_DISPATCH_NOTIFY | ACPI_GPE_LEVEL_TRIGGERED);
            GpeEventInfo->Dispatch.DeviceNode = DeviceNode;
    }

    /*
     * If we already have an implicit notify on this GPE, add
     * this device to the notify list.
     */
    if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
            ACPI_GPE_DISPATCH_NOTIFY)
    {
        /* Ensure that the device is not already in the list */

        Notify = GpeEventInfo->Dispatch.NotifyList;
        while (Notify)
        {
            if (Notify->DeviceNode == DeviceNode)
            {
                Status = AE_ALREADY_EXISTS;
                goto UnlockAndExit;
            }
            Notify = Notify->Next;
        }

        /* Add this device to the notify list for this GPE */

        NewNotify->DeviceNode = DeviceNode;
        NewNotify->Next = GpeEventInfo->Dispatch.NotifyList;
```



```
        GpeEventInfo->Dispatch.NotifyList = NewNotify;
        NewNotify = NULL;
    }

    /* Mark the GPE as a possible wake event */

    GpeEventInfo->Flags |= ACPI_GPE_CAN_WAKE;
    Status = AE_OK;
    }


UnlockAndExit:
    AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);

    /* Delete the notify object if it was not used above */

    if (NewNotify)
    {
        ACPI_FREE (NewNotify);
    }
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiSetupGpeForWake)


/*******************************************************************************
 *
 * FUNCTION:    AcpiSetGpeWakeMask
 *
 * PARAMETERS:  GpeDevice           - Parent GPE Device. NULL for GPE0/GPE1
 *              GpeNumber           - GPE level within the GPE block
 *              Action              - Enable or Disable
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Set or clear the GPE's wakeup enable mask bit. The GPE must
 *              already be marked as a WAKE GPE.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiSetGpeWakeMask (
    ACPI_HANDLE             GpeDevice,
    UINT32                  GpeNumber,
    UINT8                   Action)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_GPE_EVENT_INFO     *GpeEventInfo;
    ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
    ACPI_CPU_FLAGS          Flags;
    UINT32                  RegisterBit;


    ACPI_FUNCTION_TRACE (AcpiSetGpeWakeMask);


    Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

    /*
     * Ensure that we have a valid GPE number and that this GPE is in
     * fact a wake GPE
     */
    GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
    if (!GpeEventInfo)
    {
```

```
470         Status = AE_BAD_PARAMETER;
471         goto UnlockAndExit;
472     }

474     if (!(GpeEventInfo->Flags & ACPI_GPE_CAN_WAKE))
475     {
476         Status = AE_TYPE;
477         goto UnlockAndExit;
478     }

480     GpeRegisterInfo = GpeEventInfo->RegisterInfo;
481     if (!GpeRegisterInfo)
482     {
483         Status = AE_NOT_EXIST;
484         goto UnlockAndExit;
485     }

487     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo);
424     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);

489     /* Perform the action */

491     switch (Action)
492     {
493     case ACPI_GPE_ENABLE:

495         ACPI_SET_BIT (GpeRegisterInfo->EnableForWake, (UINT8) RegisterBit);
496         break;

498     case ACPI_GPE_DISABLE:

500         ACPI_CLEAR_BIT (GpeRegisterInfo->EnableForWake, (UINT8) RegisterBit);
501         break;

503     default:

505         ACPI_ERROR ((AE_INFO, "%u, Invalid action", Action));
506         Status = AE_BAD_PARAMETER;
507         break;
508     }

510 UnlockAndExit:
511     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
512     return_ACPI_STATUS (Status);
513 }
```

unchanged_portion_omitted_

```
739 ACPI_EXPORT_SYMBOL (AcpiEnableAllRuntimeGpes)


742 /******************************************************************************
743  *
744  * FUNCTION:    AcpiInstallGpeBlock
745  *
746  * PARAMETERS:  GpeDevice           - Handle to the parent GPE Block Device
747  *              GpeBlockAddress     - Address and SpaceID
748  *              RegisterCount       - Number of GPE register pairs in the block
749  *              InterruptNumber     - H/W interrupt for the block
750  *
751  * RETURN:      Status
752  *
753  * DESCRIPTION: Create and Install a block of GPE registers. The GPEs are not
754  *              enabled here.
755  *
756  ******************************************************************************/
```



```
758 ACPI_STATUS
759 AcpiInstallGpeBlock (
760     ACPI_HANDLE             GpeDevice,
761     ACPI_GENERIC_ADDRESS    *GpeBlockAddress,
762     UINT32                  RegisterCount,
763     UINT32                  InterruptNumber)
764 {
765     ACPI_STATUS             Status;
766     ACPI_OPERAND_OBJECT     *ObjDesc;
767     ACPI_NAMESPACE_NODE     *Node;
768     ACPI_GPE_BLOCK_INFO     *GpeBlock;


771     ACPI_FUNCTION_TRACE (AcpiInstallGpeBlock);


774     if ((!GpeDevice)       ||
775         (!GpeBlockAddress) ||
776         (!RegisterCount))
777     {
778         return_ACPI_STATUS (AE_BAD_PARAMETER);
779     }

781     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
782     if (ACPI_FAILURE (Status))
783     {
784         return_ACPI_STATUS (Status);
718         return (Status);
785     }

787     Node = AcpiNsValidateHandle (GpeDevice);
788     if (!Node)
789     {
790         Status = AE_BAD_PARAMETER;
791         goto UnlockAndExit;
792     }

794     /*
795      * For user-installed GPE Block Devices, the GpeBlockBaseNumber
796      * is always zero
797      */
798     Status = AcpiEvCreateGpeBlock (Node, GpeBlockAddress, RegisterCount,
799                 0, InterruptNumber, &GpeBlock);
800     if (ACPI_FAILURE (Status))
801     {
802         goto UnlockAndExit;
803     }

805     /* Install block in the DeviceObject attached to the node */

807     ObjDesc = AcpiNsGetAttachedObject (Node);
808     if (!ObjDesc)
809     {
810         /*
811          * No object, create a new one (Device nodes do not always have
812          * an attached object)
813          */
814         ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_DEVICE);
815         if (!ObjDesc)
816         {
817             Status = AE_NO_MEMORY;
818             goto UnlockAndExit;
819         }

821         Status = AcpiNsAttachObject (Node, ObjDesc, ACPI_TYPE_DEVICE);
```

```
823         /* Remove local reference to the object */

825         AcpiUtRemoveReference (ObjDesc);
826         if (ACPI_FAILURE (Status))
827         {
828             goto UnlockAndExit;
829         }
830     }

832     /* Now install the GPE block in the DeviceObject */

834     ObjDesc->Device.GpeBlock = GpeBlock;


837 UnlockAndExit:
838     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
839     return_ACPI_STATUS (Status);
840 }

842 ACPI_EXPORT_SYMBOL (AcpiInstallGpeBlock)


845 /*******************************************************************************
846  *
847  * FUNCTION:    AcpiRemoveGpeBlock
848  *
849  * PARAMETERS:  GpeDevice           - Handle to the parent GPE Block Device
850  *
851  * RETURN:      Status
852  *
853  * DESCRIPTION: Remove a previously installed block of GPE registers
854  *
855  ******************************************************************************/

857 ACPI_STATUS
858 AcpiRemoveGpeBlock (
859     ACPI_HANDLE             GpeDevice)
860 {
861     ACPI_OPERAND_OBJECT     *ObjDesc;
862     ACPI_STATUS             Status;
863     ACPI_NAMESPACE_NODE     *Node;


866     ACPI_FUNCTION_TRACE (AcpiRemoveGpeBlock);


869     if (!GpeDevice)
870     {
871         return_ACPI_STATUS (AE_BAD_PARAMETER);
872     }

874     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
875     if (ACPI_FAILURE (Status))
876     {
877         return_ACPI_STATUS (Status);
878         return (Status);
879     }

880     Node = AcpiNsValidateHandle (GpeDevice);
881     if (!Node)
882     {
883         Status = AE_BAD_PARAMETER;
884         goto UnlockAndExit;
885     }

887     /* Get the DeviceObject attached to the node */
```



```
889     ObjDesc = AcpiNsGetAttachedObject (Node);
890     if (!ObjDesc ||
891         !ObjDesc->Device.GpeBlock)
892     {
893         return_ACPI_STATUS (AE_NULL_OBJECT);
894     }

896     /* Delete the GPE block (but not the DeviceObject) */

898     Status = AcpiEvDeleteGpeBlock (ObjDesc->Device.GpeBlock);
899     if (ACPI_SUCCESS (Status))
900     {
901         ObjDesc->Device.GpeBlock = NULL;
902     }

904 UnlockAndExit:
905     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
906     return_ACPI_STATUS (Status);
907 }
______unchanged_portion_omitted_

966 ACPI_EXPORT_SYMBOL (AcpiGetGpeDevice)

968 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   10049 Thu Dec 26 13:49:00 2013
new/usr/src/common/acpica/components/events/evxfregn.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: evxfregn - External Interfaces, ACPI Operation Regions and
 *                         Address Spaces.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EVXFREGN_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acevents.h"

#define _COMPONENT          ACPI_EVENTS
        ACPI_MODULE_NAME    ("evxfregn")


/******************************************************************************
 *
```



```
 * FUNCTION:    AcpiInstallAddressSpaceHandler
 *
 * PARAMETERS:  Device          - Handle for the device
 *              SpaceId         - The address space ID
 *              Handler         - Address of the handler
 *              Setup           - Address of the setup function
 *              Context         - Value passed to the handler on each access
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Install a handler for all OpRegions of a given SpaceId.
 *
 * NOTE: This function should only be called after AcpiEnableSubsystem has
 * been called. This is because any _REG methods associated with the Space ID
 * are executed here, and these methods can only be safely executed after
 * the default handlers have been installed and the hardware has been
 * initialized (via AcpiEnableSubsystem.)
 *
 *****************************************************************************/

ACPI_STATUS
AcpiInstallAddressSpaceHandler (
    ACPI_HANDLE             Device,
    ACPI_ADR_SPACE_TYPE     SpaceId,
    ACPI_ADR_SPACE_HANDLER  Handler,
    ACPI_ADR_SPACE_SETUP    Setup,
    void                    *Context)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiInstallAddressSpaceHandler);


    /* Parameter validation */

    if (!Device)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Convert and validate the device handle */

    Node = AcpiNsValidateHandle (Device);
    if (!Node)
    {
        Status = AE_BAD_PARAMETER;
        goto UnlockAndExit;
    }

    /* Install the handler for all Regions for this Space ID */

    Status = AcpiEvInstallSpaceHandler (Node, SpaceId, Handler, Setup, Context);
    if (ACPI_FAILURE (Status))
    {
        goto UnlockAndExit;
    }

    /*
```

```
125      * For the default SpaceIDs, (the IDs for which there are default region han
126      * installed) Only execute the _REG methods if the global initialization _RE
127      * methods have already been run (via AcpiInitializeObjects). In other words
128      * we will defer the execution of the _REG methods for these SpaceIDs until
129      * execution of AcpiInitializeObjects. This is done because we need the hand
130      * for the default spaces (mem/io/pci/table) to be installed before we can r
131      * any control methods (or _REG methods). There is known BIOS code that depe
132      * on this.
133      *
134      * For all other SpaceIDs, we can safely execute the _REG methods immediatel
135      * This means that for IDs like EmbeddedController, this function should be
136      * only after AcpiEnableSubsystem has been called.
137      */
138     switch (SpaceId)
139     {
140     case ACPI_ADR_SPACE_SYSTEM_MEMORY:
141     case ACPI_ADR_SPACE_SYSTEM_IO:
142     case ACPI_ADR_SPACE_PCI_CONFIG:
143     case ACPI_ADR_SPACE_DATA_TABLE:

145         if (!AcpiGbl_RegMethodsExecuted)
146         {
147             /* We will defer execution of the _REG methods for this space */
148             goto UnlockAndExit;
149         }
150         break;

152     default:

154         break;
155     }

157     /* Run all _REG methods for this address space */

159     Status = AcpiEvExecuteRegMethods (Node, SpaceId);


162 UnlockAndExit:
163     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
164     return_ACPI_STATUS (Status);
165 }
```

*unchanged_portion_omitted_*

```
314 ACPI_EXPORT_SYMBOL (AcpiRemoveAddressSpaceHandler)
```

```
**********************************************************
   20453 Thu Dec 26 13:49:01 2013
new/usr/src/common/acpica/components/executer/exconfig.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exconfig - Namespace reconfiguration (Load/Unload opcodes)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXCONFIG_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acinterp.h"
  49 #include "acnamesp.h"
  50 #include "actables.h"
  51 #include "acdispat.h"
  52 #include "acevents.h"
  53 #include "amlcode.h"


  56 #define _COMPONENT          ACPI_EXECUTER
  57         ACPI_MODULE_NAME    ("exconfig")

  59 /* Local prototypes */
```



```
  61 static ACPI_STATUS
  62 AcpiExAddTable (
  63     UINT32                  TableIndex,
  64     ACPI_NAMESPACE_NODE     *ParentNode,
  65     ACPI_OPERAND_OBJECT     **DdbHandle);

  67 static ACPI_STATUS
  68 AcpiExRegionRead (
  69     ACPI_OPERAND_OBJECT     *ObjDesc,
  70     UINT32                  Length,
  71     UINT8                   *Buffer);


  74 /*******************************************************************************
  75  *
  76  * FUNCTION:    AcpiExAddTable
  77  *
  78  * PARAMETERS:  Table               - Pointer to raw table
  79  *              ParentNode          - Where to load the table (scope)
  80  *              DdbHandle           - Where to return the table handle.
  81  *
  82  * RETURN:      Status
  83  *
  84  * DESCRIPTION: Common function to Install and Load an ACPI table with a
  85  *              returned table handle.
  86  *
  87  ******************************************************************************/

  89 static ACPI_STATUS
  90 AcpiExAddTable (
  91     UINT32                  TableIndex,
  92     ACPI_NAMESPACE_NODE     *ParentNode,
  93     ACPI_OPERAND_OBJECT     **DdbHandle)
  94 {
  95     ACPI_OPERAND_OBJECT     *ObjDesc;
  96     ACPI_STATUS             Status;
  97     ACPI_OWNER_ID           OwnerId;


 100     ACPI_FUNCTION_TRACE (ExAddTable);


 103     /* Create an object to be the table handle */

 105     ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_LOCAL_REFERENCE);
 106     if (!ObjDesc)
 107     {
 108         return_ACPI_STATUS (AE_NO_MEMORY);
 109     }

 111     /* Init the table handle */

 113     ObjDesc->Common.Flags |= AOPOBJ_DATA_VALID;
 114     ObjDesc->Reference.Class = ACPI_REFCLASS_TABLE;
 115     *DdbHandle = ObjDesc;

 117     /* Install the new table into the local data structures */

 119     ObjDesc->Reference.Value = TableIndex;

 121     /* Add the table to the namespace */

 123     Status = AcpiNsLoadTable (TableIndex, ParentNode);
 124     if (ACPI_FAILURE (Status))
 125     {
```

```
126         AcpiUtRemoveReference (ObjDesc);
127         *DdbHandle = NULL;
128         return_ACPI_STATUS (Status);
129     }

131     /* Execute any module-level code that was found in the table */

133     AcpiExExitInterpreter ();
134     AcpiNsExecModuleCodeList ();
135     AcpiExEnterInterpreter ();

137     /*
138      * Update GPEs for any new _Lxx/_Exx methods. Ignore errors. The host is
139      * responsible for discovering any new wake GPEs by running _PRW methods
140      * that may have been loaded by this table.
141      */
142     Status = AcpiTbGetOwnerId (TableIndex, &OwnerId);
143     if (ACPI_SUCCESS (Status))
144     {
145         AcpiEvUpdateGpes (OwnerId);
146     }

148     return_ACPI_STATUS (AE_OK);
149 }


152 /*******************************************************************************
153  *
154  * FUNCTION:    AcpiExLoadTableOp
155  *
156  * PARAMETERS:  WalkState           - Current state with operands
157  *              ReturnDesc          - Where to store the return object
158  *
159  * RETURN:      Status
160  *
161  * DESCRIPTION: Load an ACPI table from the RSDT/XSDT
162  *
163  ******************************************************************************/

165 ACPI_STATUS
166 AcpiExLoadTableOp (
167     ACPI_WALK_STATE         *WalkState,
168     ACPI_OPERAND_OBJECT     **ReturnDesc)
169 {
170     ACPI_STATUS             Status;
171     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
172     ACPI_NAMESPACE_NODE     *ParentNode;
173     ACPI_NAMESPACE_NODE     *StartNode;
174     ACPI_NAMESPACE_NODE     *ParameterNode = NULL;
175     ACPI_OPERAND_OBJECT     *DdbHandle;
176     ACPI_TABLE_HEADER       *Table;
177     UINT32                  TableIndex;


180     ACPI_FUNCTION_TRACE (ExLoadTableOp);


183     /* Validate lengths for the Signature, OemId, and OemTableId strings */
182     /* Validate lengths for the SignatureString, OEMIDString, OEMTableID */

185     if ((Operand[0]->String.Length > ACPI_NAME_SIZE) ||
186         (Operand[1]->String.Length > ACPI_OEM_ID_SIZE) ||
187         (Operand[2]->String.Length > ACPI_OEM_TABLE_ID_SIZE))
188     {
189         return_ACPI_STATUS (AE_AML_STRING_LIMIT);
188         return_ACPI_STATUS (AE_BAD_PARAMETER);
```



```
190     }

192     /* Find the ACPI table in the RSDT/XSDT */

194     Status = AcpiTbFindTable (
195                 Operand[0]->String.Pointer,
193     Status = AcpiTbFindTable (Operand[0]->String.Pointer,
196                 Operand[1]->String.Pointer,
197                 Operand[2]->String.Pointer, &TableIndex);
198     if (ACPI_FAILURE (Status))
199     {
200         if (Status != AE_NOT_FOUND)
201         {
202             return_ACPI_STATUS (Status);
203         }

205         /* Table not found, return an Integer=0 and AE_OK */

207         DdbHandle = AcpiUtCreateIntegerObject ((UINT64) 0);
208         if (!DdbHandle)
209         {
210             return_ACPI_STATUS (AE_NO_MEMORY);
211         }

213         *ReturnDesc = DdbHandle;
214         return_ACPI_STATUS (AE_OK);
215     }

217     /* Default nodes */

219     StartNode = WalkState->ScopeInfo->Scope.Node;
220     ParentNode = AcpiGbl_RootNode;

222     /* RootPath (optional parameter) */

224     if (Operand[3]->String.Length > 0)
225     {
226         /*
227          * Find the node referenced by the RootPathString. This is the
228          * location within the namespace where the table will be loaded.
229          */
230         Status = AcpiNsGetNode (StartNode, Operand[3]->String.Pointer,
231                     ACPI_NS_SEARCH_PARENT, &ParentNode);
232         if (ACPI_FAILURE (Status))
233         {
234             return_ACPI_STATUS (Status);
235         }
236     }

238     /* ParameterPath (optional parameter) */

240     if (Operand[4]->String.Length > 0)
241     {
242         if ((Operand[4]->String.Pointer[0] != AML_ROOT_PREFIX) &&
243             (Operand[4]->String.Pointer[0] != AML_PARENT_PREFIX))
240         if ((Operand[4]->String.Pointer[0] != '\\') &&
241             (Operand[4]->String.Pointer[0] != '^'))
244         {
245             /*
246              * Path is not absolute, so it will be relative to the node
247              * referenced by the RootPathString (or the NS root if omitted)
248              */
249             StartNode = ParentNode;
250         }

252         /* Find the node referenced by the ParameterPathString */
```

```
254         Status = AcpiNsGetNode (StartNode, Operand[4]->String.Pointer,
255                     ACPI_NS_SEARCH_PARENT, &ParameterNode);
256         if (ACPI_FAILURE (Status))
257         {
258             return_ACPI_STATUS (Status);
259         }
260     }

262     /* Load the table into the namespace */

264     Status = AcpiExAddTable (TableIndex, ParentNode, &DdbHandle);
265     if (ACPI_FAILURE (Status))
266     {
267         return_ACPI_STATUS (Status);
268     }

270     /* Parameter Data (optional) */

272     if (ParameterNode)
273     {
274         /* Store the parameter data into the optional parameter object */

276         Status = AcpiExStore (Operand[5],
277                     ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, ParameterNode),
278                     WalkState);
279         if (ACPI_FAILURE (Status))
280         {
281             (void) AcpiExUnloadTable (DdbHandle);

283             AcpiUtRemoveReference (DdbHandle);
284             return_ACPI_STATUS (Status);
285         }
286     }

288     Status = AcpiGetTableByIndex (TableIndex, &Table);
289     if (ACPI_SUCCESS (Status))
290     {
291         ACPI_INFO ((AE_INFO, "Dynamic OEM Table Load:"));
292         AcpiTbPrintTableHeader (0, Table);
293     }

295     /* Invoke table handler if present */

297     if (AcpiGbl_TableHandler)
298     {
299         (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_LOAD, Table,
300                     AcpiGbl_TableHandlerContext);
301     }

303     *ReturnDesc = DdbHandle;
304     return_ACPI_STATUS (Status);
305 }


308 /*****************************************************************************
309  *
310  * FUNCTION:    AcpiExRegionRead
311  *
312  * PARAMETERS:  ObjDesc         - Region descriptor
313  *              Length          - Number of bytes to read
314  *              Buffer          - Pointer to where to put the data
315  *
316  * RETURN:      Status
317  *
318  * DESCRIPTION: Read data from an operation region. The read starts from the
```



```
319  *              beginning of the region.
320  *
321  *****************************************************************************/

323 static ACPI_STATUS
324 AcpiExRegionRead (
325     ACPI_OPERAND_OBJECT     *ObjDesc,
326     UINT32                  Length,
327     UINT8                   *Buffer)
328 {
329     ACPI_STATUS             Status;
330     UINT64                  Value;
331     UINT32                  RegionOffset = 0;
332     UINT32                  i;


335     /* Bytewise reads */

337     for (i = 0; i < Length; i++)
338     {
339         Status = AcpiEvAddressSpaceDispatch (ObjDesc, NULL, ACPI_READ,
337         Status = AcpiEvAddressSpaceDispatch (ObjDesc, ACPI_READ,
340                     RegionOffset, 8, &Value);
341         if (ACPI_FAILURE (Status))
342         {
343             return (Status);
344         }

346         *Buffer = (UINT8) Value;
347         Buffer++;
348         RegionOffset++;
349     }

351     return (AE_OK);
352 }


355 /*****************************************************************************
356  *
357  * FUNCTION:    AcpiExLoadOp
358  *
359  * PARAMETERS:  ObjDesc         - Region or Buffer/Field where the table will be
360  *                                obtained
361  *              Target          - Where a handle to the table will be stored
362  *              WalkState       - Current state
363  *
364  * RETURN:      Status
365  *
366  * DESCRIPTION: Load an ACPI table from a field or operation region
367  *
368  * NOTE: Region Fields (Field, BankField, IndexFields) are resolved to buffer
369  *       objects before this code is reached.
370  *
371  *       If source is an operation region, it must refer to SystemMemory, as
372  *       per the ACPI specification.
373  *
374  *****************************************************************************/

376 ACPI_STATUS
377 AcpiExLoadOp (
378     ACPI_OPERAND_OBJECT     *ObjDesc,
379     ACPI_OPERAND_OBJECT     *Target,
380     ACPI_WALK_STATE         *WalkState)
381 {
382     ACPI_OPERAND_OBJECT     *DdbHandle;
383     ACPI_TABLE_HEADER       *Table;
```

```
384     ACPI_TABLE_DESC         TableDesc;
385     UINT32                  TableIndex;
386     ACPI_STATUS             Status;
387     UINT32                  Length;


390     ACPI_FUNCTION_TRACE (ExLoadOp);


393     ACPI_MEMSET (&TableDesc, 0, sizeof (ACPI_TABLE_DESC));

395     /* Source Object can be either an OpRegion or a Buffer/Field */

397     switch (ObjDesc->Common.Type)
398     {
399     case ACPI_TYPE_REGION:

401         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
402             "Load table from Region %p\n", ObjDesc));

404         /* Region must be SystemMemory (from ACPI spec) */

406         if (ObjDesc->Region.SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY)
407         {
408             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
409         }

411         /*
412          * If the Region Address and Length have not been previously evaluated,
413          * evaluate them now and save the results.
414          */
415         if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
416         {
417             Status = AcpiDsGetRegionArguments (ObjDesc);
418             if (ACPI_FAILURE (Status))
419             {
420                 return_ACPI_STATUS (Status);
421             }
422         }

424         /* Get the table header first so we can get the table length */

426         Table = ACPI_ALLOCATE (sizeof (ACPI_TABLE_HEADER));
427         if (!Table)
428         {
429             return_ACPI_STATUS (AE_NO_MEMORY);
430         }

432         Status = AcpiExRegionRead (ObjDesc, sizeof (ACPI_TABLE_HEADER),
433                     ACPI_CAST_PTR (UINT8, Table));
434         Length = Table->Length;
435         ACPI_FREE (Table);

437         if (ACPI_FAILURE (Status))
438         {
439             return_ACPI_STATUS (Status);
440         }

442         /* Must have at least an ACPI table header */

444         if (Length < sizeof (ACPI_TABLE_HEADER))
445         {
446             return_ACPI_STATUS (AE_INVALID_TABLE_LENGTH);
447         }

449         /*
```



```
450          * The original implementation simply mapped the table, with no copy.
451          * However, the memory region is not guaranteed to remain stable and
452          * we must copy the table to a local buffer. For example, the memory
453          * region is corrupted after suspend on some machines. Dynamically
454          * loaded tables are usually small, so this overhead is minimal.
455          *
456          * The latest implementation (5/2009) does not use a mapping at all.
457          * We use the low-level operation region interface to read the table
458          * instead of the obvious optimization of using a direct mapping.
459          * This maintains a consistent use of operation regions across the
460          * entire subsystem. This is important if additional processing must
461          * be performed in the (possibly user-installed) operation region
462          * handler. For example, AcpiExec and ASLTS depend on this.
463          */

465         /* Allocate a buffer for the table */

467         TableDesc.Pointer = ACPI_ALLOCATE (Length);
468         if (!TableDesc.Pointer)
469         {
470             return_ACPI_STATUS (AE_NO_MEMORY);
471         }

473         /* Read the entire table */

475         Status = AcpiExRegionRead (ObjDesc, Length,
476                     ACPI_CAST_PTR (UINT8, TableDesc.Pointer));
477         if (ACPI_FAILURE (Status))
478         {
479             ACPI_FREE (TableDesc.Pointer);
480             return_ACPI_STATUS (Status);
481         }

483         TableDesc.Address = ObjDesc->Region.Address;
484         break;


486     case ACPI_TYPE_BUFFER: /* Buffer or resolved RegionField */

488         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
489             "Load table from Buffer or Field %p\n", ObjDesc));

491         /* Must have at least an ACPI table header */

493         if (ObjDesc->Buffer.Length < sizeof (ACPI_TABLE_HEADER))
494         {
495             return_ACPI_STATUS (AE_INVALID_TABLE_LENGTH);
496         }

498         /* Get the actual table length from the table header */

500         Table = ACPI_CAST_PTR (ACPI_TABLE_HEADER, ObjDesc->Buffer.Pointer);
501         Length = Table->Length;

503         /* Table cannot extend beyond the buffer */

505         if (Length > ObjDesc->Buffer.Length)
506         {
507             return_ACPI_STATUS (AE_AML_BUFFER_LIMIT);
508         }
509         if (Length < sizeof (ACPI_TABLE_HEADER))
510         {
511             return_ACPI_STATUS (AE_INVALID_TABLE_LENGTH);
512         }

514         /*
```

```
 515          * Copy the table from the buffer because the buffer could be modified
 516          * or even deleted in the future
 517          */
 518         TableDesc.Pointer = ACPI_ALLOCATE (Length);
 519         if (!TableDesc.Pointer)
 520         {
 521             return_ACPI_STATUS (AE_NO_MEMORY);
 522         }

 524         ACPI_MEMCPY (TableDesc.Pointer, Table, Length);
 525         TableDesc.Address = ACPI_TO_INTEGER (TableDesc.Pointer);
 526         break;

 528     default:

 528     default:
 530         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
 531     }

 533     /* Validate table checksum (will not get validated in TbAddTable) */

 535     Status = AcpiTbVerifyChecksum (TableDesc.Pointer, Length);
 536     if (ACPI_FAILURE (Status))
 537     {
 538         ACPI_FREE (TableDesc.Pointer);
 539         return_ACPI_STATUS (Status);
 540     }

 542     /* Complete the table descriptor */

 544     TableDesc.Length = Length;
 545     TableDesc.Flags = ACPI_TABLE_ORIGIN_ALLOCATED;

 547     /* Install the new table into the local data structures */

 549     Status = AcpiTbAddTable (&TableDesc, &TableIndex);
 550     if (ACPI_FAILURE (Status))
 551     {
 552         /* Delete allocated table buffer */

 554         AcpiTbDeleteTable (&TableDesc);
 555         return_ACPI_STATUS (Status);
 556     }

 558     /*
 559      * Add the table to the namespace.
 560      *
 561      * Note: Load the table objects relative to the root of the namespace.
 562      * This appears to go against the ACPI specification, but we do it for
 563      * compatibility with other ACPI implementations.
 564      */
 565     Status = AcpiExAddTable (TableIndex, AcpiGbl_RootNode, &DdbHandle);
 566     if (ACPI_FAILURE (Status))
 567     {
 568         /* On error, TablePtr was deallocated above */

 570         return_ACPI_STATUS (Status);
 571     }

 573     /* Store the DdbHandle into the Target operand */

 575     Status = AcpiExStore (DdbHandle, Target, WalkState);
 576     if (ACPI_FAILURE (Status))
 577     {
 578         (void) AcpiExUnloadTable (DdbHandle);
```



```
 580         /* TablePtr was deallocated above */

 582         AcpiUtRemoveReference (DdbHandle);
 583         return_ACPI_STATUS (Status);
 584     }

 586     ACPI_INFO ((AE_INFO, "Dynamic OEM Table Load:"));
 587     AcpiTbPrintTableHeader (0, TableDesc.Pointer);

 589     /* Remove the reference by added by AcpiExStore above */

 591     AcpiUtRemoveReference (DdbHandle);

 593     /* Invoke table handler if present */

 595     if (AcpiGbl_TableHandler)
 596     {
 597         (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_LOAD, TableDesc.Pointer,
 598                     AcpiGbl_TableHandlerContext);
 599     }

 601     return_ACPI_STATUS (Status);
 602 }


 605 /*******************************************************************************
 606  *
 607  * FUNCTION:    AcpiExUnloadTable
 608  *
 609  * PARAMETERS:  DdbHandle           - Handle to a previously loaded table
 610  *
 611  * RETURN:      Status
 612  *
 613  * DESCRIPTION: Unload an ACPI table
 614  *
 615  ******************************************************************************/

 617 ACPI_STATUS
 618 AcpiExUnloadTable (
 619     ACPI_OPERAND_OBJECT     *DdbHandle)
 620 {
 621     ACPI_STATUS             Status = AE_OK;
 622     ACPI_OPERAND_OBJECT     *TableDesc = DdbHandle;
 623     UINT32                  TableIndex;
 624     ACPI_TABLE_HEADER       *Table;


 627     ACPI_FUNCTION_TRACE (ExUnloadTable);


 630     /*
 631      * Validate the handle
 632      * Although the handle is partially validated in AcpiExReconfiguration()
 633      * when it calls AcpiExResolveOperands(), the handle is more completely
 634      * validated here.
 635      *
 636      * Handle must be a valid operand object of type reference. Also, the
 637      * DdbHandle must still be marked valid (table has not been previously
 638      * unloaded)
 639      */
 640     if ((!DdbHandle) ||
 641         (ACPI_GET_DESCRIPTOR_TYPE (DdbHandle) != ACPI_DESC_TYPE_OPERAND) ||
 642         (DdbHandle->Common.Type != ACPI_TYPE_LOCAL_REFERENCE) ||
 643         (!(DdbHandle->Common.Flags & AOPOBJ_DATA_VALID)))
 644     {
 645         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
```

```
644         return_ACPI_STATUS (AE_BAD_PARAMETER);
646     }

648     /* Get the table index from the DdbHandle */

650     TableIndex = TableDesc->Reference.Value;

652     /* Ensure the table is still loaded */

654     if (!AcpiTbIsTableLoaded (TableIndex))
655     {
656         return_ACPI_STATUS (AE_NOT_EXIST);
657     }

659     /* Invoke table handler if present */

661     if (AcpiGbl_TableHandler)
662     {
663         Status = AcpiGetTableByIndex (TableIndex, &Table);
664         if (ACPI_SUCCESS (Status))
665         {
666             (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_UNLOAD, Table,
667                         AcpiGbl_TableHandlerContext);
668         }
669     }

671     /* Delete the portion of the namespace owned by this table */

673     Status = AcpiTbDeleteNamespaceByOwner (TableIndex);
674     if (ACPI_FAILURE (Status))
675     {
676         return_ACPI_STATUS (Status);
677     }

679     (void) AcpiTbReleaseOwnerId (TableIndex);
680     AcpiTbSetTableLoadedFlag (TableIndex, FALSE);

682     /*
683      * Invalidate the handle. We do this because the handle may be stored
684      * in a named object and may not be actually deleted until much later.
685      */
686     DdbHandle->Common.Flags &= ~AOPOBJ_DATA_VALID;
687     return_ACPI_STATUS (AE_OK);
688 }
```

****
21068 Thu Dec 26 13:49:01 2013
new/usr/src/common/acpica/components/executer/exconvrt.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exconvrt - Object conversion routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EXCONVRT_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exconvrt")

/* Local prototypes */

static UINT32
AcpiExConvertToAscii (
```



```
    UINT64                  Integer,
    UINT16                  Base,
    UINT8                   *String,
    UINT8                   MaxLength);


/******************************************************************************
 *
 * FUNCTION:    AcpiExConvertToInteger
 *
 * PARAMETERS:  ObjDesc         - Object to be converted. Must be an
 *                                Integer, Buffer, or String
 *              ResultDesc      - Where the new Integer object is returned
 *              Flags           - Used for string conversion
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert an ACPI Object to an integer.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiExConvertToInteger (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     **ResultDesc,
    UINT32                  Flags)
{
    ACPI_OPERAND_OBJECT     *ReturnDesc;
    UINT8                   *Pointer;
    UINT64                  Result;
    UINT32                  i;
    UINT32                  Count;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE_PTR (ExConvertToInteger, ObjDesc);


    switch (ObjDesc->Common.Type)
    {
    case ACPI_TYPE_INTEGER:

        /* No conversion necessary */

        *ResultDesc = ObjDesc;
        return_ACPI_STATUS (AE_OK);

    case ACPI_TYPE_BUFFER:
    case ACPI_TYPE_STRING:

        /* Note: Takes advantage of common buffer/string fields */

        Pointer = ObjDesc->Buffer.Pointer;
        Count   = ObjDesc->Buffer.Length;
        break;

    default:

        return_ACPI_STATUS (AE_TYPE);
    }

    /*
     * Convert the buffer/string to an integer. Note that both buffers and
     * strings are treated as raw data - we don't convert ascii to hex for
     * strings.
     *
```

```
126      * There are two terminating conditions for the loop:
127      * 1) The size of an integer has been reached, or
128      * 2) The end of the buffer or string has been reached
129      */
130     Result = 0;

132     /* String conversion is different than Buffer conversion */

134     switch (ObjDesc->Common.Type)
135     {
136     case ACPI_TYPE_STRING:

137         /*
138          * Convert string to an integer - for most cases, the string must be
139          * hexadecimal as per the ACPI specification. The only exception (as
140          * of ACPI 3.0) is that the ToInteger() operator allows both decimal
141          * and hexadecimal strings (hex prefixed with "0x").
142          */
143         Status = AcpiUtStrtoul64 ((char *) Pointer, Flags, &Result);
144         if (ACPI_FAILURE (Status))
145         {
146             return_ACPI_STATUS (Status);
147         }
148         break;


150     case ACPI_TYPE_BUFFER:

152         /* Check for zero-length buffer */

154         if (!Count)
155         {
156             return_ACPI_STATUS (AE_AML_BUFFER_LIMIT);
157         }

159         /* Transfer no more than an integer's worth of data */

161         if (Count > AcpiGbl_IntegerByteWidth)
162         {
163             Count = AcpiGbl_IntegerByteWidth;
164         }

166         /*
167          * Convert buffer to an integer - we simply grab enough raw data
168          * from the buffer to fill an integer
169          */
170         for (i = 0; i < Count; i++)
171         {
172             /*
173              * Get next byte and shift it into the Result.
174              * Little endian is used, meaning that the first byte of the buffer
175              * is the LSB of the integer
176              */
177             Result |= (((UINT64) Pointer[i]) << (i * 8));
178         }
179         break;


181     default:

183         /* No other types can get here */

185         break;
186     }

188     /* Create a new integer */
```



```
190     ReturnDesc = AcpiUtCreateIntegerObject (Result);
191     if (!ReturnDesc)
192     {
193         return_ACPI_STATUS (AE_NO_MEMORY);
194     }

196     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Converted value: %8.8X%8.8X\n",
197         ACPI_FORMAT_UINT64 (Result)));

199     /* Save the Result */

201     (void) AcpiExTruncateFor32bitTable (ReturnDesc);
202     AcpiExTruncateFor32bitTable (ReturnDesc);
202     *ResultDesc = ReturnDesc;
203     return_ACPI_STATUS (AE_OK);
204 }


207 /*******************************************************************************
208  *
209  * FUNCTION:    AcpiExConvertToBuffer
210  *
211  * PARAMETERS:  ObjDesc         - Object to be converted. Must be an
212  *                                Integer, Buffer, or String
213  *              ResultDesc      - Where the new buffer object is returned
214  *
215  * RETURN:      Status
216  *
217  * DESCRIPTION: Convert an ACPI Object to a Buffer
218  *
219  ******************************************************************************/

221 ACPI_STATUS
222 AcpiExConvertToBuffer (
223     ACPI_OPERAND_OBJECT     *ObjDesc,
224     ACPI_OPERAND_OBJECT     **ResultDesc)
225 {
226     ACPI_OPERAND_OBJECT     *ReturnDesc;
227     UINT8                   *NewBuf;


230     ACPI_FUNCTION_TRACE_PTR (ExConvertToBuffer, ObjDesc);


233     switch (ObjDesc->Common.Type)
234     {
235     case ACPI_TYPE_BUFFER:

237         /* No conversion necessary */

239         *ResultDesc = ObjDesc;
240         return_ACPI_STATUS (AE_OK);


243     case ACPI_TYPE_INTEGER:

244         /*
245          * Create a new Buffer object.
246          * Need enough space for one integer
247          */
248         ReturnDesc = AcpiUtCreateBufferObject (AcpiGbl_IntegerByteWidth);
249         if (!ReturnDesc)
250         {
251             return_ACPI_STATUS (AE_NO_MEMORY);
252         }
```

```
254         /* Copy the integer to the buffer, LSB first */

256         NewBuf = ReturnDesc->Buffer.Pointer;
257         ACPI_MEMCPY (NewBuf,
258                         &ObjDesc->Integer.Value,
259                         AcpiGbl_IntegerByteWidth);
260         break;


262     case ACPI_TYPE_STRING:

263         /*
264          * Create a new Buffer object
265          * Size will be the string length
266          *
267          * NOTE: Add one to the string length to include the null terminator.
268          * The ACPI spec is unclear on this subject, but there is existing
269          * ASL/AML code that depends on the null being transferred to the new
270          * buffer.
271          */
272         ReturnDesc = AcpiUtCreateBufferObject (
273                         (ACPI_SIZE) ObjDesc->String.Length + 1);
274         if (!ReturnDesc)
275         {
276             return_ACPI_STATUS (AE_NO_MEMORY);
277         }

279         /* Copy the string to the buffer */

281         NewBuf = ReturnDesc->Buffer.Pointer;
282         ACPI_STRNCPY ((char *) NewBuf, (char *) ObjDesc->String.Pointer,
283             ObjDesc->String.Length);
284         break;

286     default:

291     default:
288         return_ACPI_STATUS (AE_TYPE);
289     }

291     /* Mark buffer initialized */

293     ReturnDesc->Common.Flags |= AOPOBJ_DATA_VALID;
294     *ResultDesc = ReturnDesc;
295     return_ACPI_STATUS (AE_OK);
296 }


299 /*******************************************************************************
300  *
301  * FUNCTION:    AcpiExConvertToAscii
302  *
303  * PARAMETERS:  Integer         - Value to be converted
304  *              Base            - ACPI_STRING_DECIMAL or ACPI_STRING_HEX
305  *              String          - Where the string is returned
306  *              DataWidth       - Size of data item to be converted, in bytes
307  *
308  * RETURN:      Actual string length
309  *
310  * DESCRIPTION: Convert an ACPI Integer to a hex or decimal string
311  *
312  ******************************************************************************/

314 static UINT32
315 AcpiExConvertToAscii (
```



```
316     UINT64                  Integer,
317     UINT16                  Base,
318     UINT8                   *String,
319     UINT8                   DataWidth)
320 {
321     UINT64                  Digit;
322     UINT32                  i;
323     UINT32                  j;
324     UINT32                  k = 0;
325     UINT32                  HexLength;
326     UINT32                  DecimalLength;
327     UINT32                  Remainder;
328     BOOLEAN                 SupressZeros;


331     ACPI_FUNCTION_ENTRY ();


334     switch (Base)
335     {
336     case 10:

338         /* Setup max length for the decimal number */

340         switch (DataWidth)
341         {
342         case 1:

344             DecimalLength = ACPI_MAX8_DECIMAL_DIGITS;
345             break;

347         case 4:

349             DecimalLength = ACPI_MAX32_DECIMAL_DIGITS;
350             break;

352         case 8:
353         default:

355             DecimalLength = ACPI_MAX64_DECIMAL_DIGITS;
356             break;
357         }

359         SupressZeros = TRUE;     /* No leading zeros */
360         Remainder = 0;

362         for (i = DecimalLength; i > 0; i--)
363         {
364             /* Divide by nth factor of 10 */

366             Digit = Integer;
367             for (j = 0; j < i; j++)
368             {
369                 (void) AcpiUtShortDivide (Digit, 10, &Digit, &Remainder);
370             }

372             /* Handle leading zeros */

374             if (Remainder != 0)
375             {
376                 SupressZeros = FALSE;
377             }

379             if (!SupressZeros)
380             {
381                 String[k] = (UINT8) (ACPI_ASCII_ZERO + Remainder);
```

```
382                 k++;
383             }
384         }
385         break;

387     case 16:

389         /* HexLength: 2 ascii hex chars per data byte */

391         HexLength = ACPI_MUL_2 (DataWidth);
392         for (i = 0, j = (HexLength-1); i < HexLength; i++, j--)
393         {
394             /* Get one hex digit, most significant digits first */

396             String[k] = (UINT8) AcpiUtHexToAsciiChar (Integer, ACPI_MUL_4 (j));
397             k++;
398         }
399         break;

401     default:
402         return (0);
403     }

405     /*
406      * Since leading zeros are suppressed, we must check for the case where
407      * the integer equals 0
408      *
409      * Finally, null terminate the string and return the length
410      */
411     if (!k)
412     {
413         String [0] = ACPI_ASCII_ZERO;
414         k = 1;
415     }

417     String [k] = 0;
418     return ((UINT32) k);
419 }


422 /*******************************************************************************
423  *
424  * FUNCTION:    AcpiExConvertToString
425  *
426  * PARAMETERS:  ObjDesc         - Object to be converted. Must be an
427  *                                Integer, Buffer, or String
428  *              ResultDesc      - Where the string object is returned
429  *              Type            - String flags (base and conversion type)
430  *
431  * RETURN:      Status
432  *
433  * DESCRIPTION: Convert an ACPI Object to a string
434  *
435  ******************************************************************************/

437 ACPI_STATUS
438 AcpiExConvertToString (
439     ACPI_OPERAND_OBJECT     *ObjDesc,
440     ACPI_OPERAND_OBJECT     **ResultDesc,
441     UINT32                  Type)
442 {
443     ACPI_OPERAND_OBJECT     *ReturnDesc;
444     UINT8                   *NewBuf;
445     UINT32                  i;
446     UINT32                  StringLength = 0;
447     UINT16                  Base = 16;
```



```
448     UINT8                   Separator = ',';


451     ACPI_FUNCTION_TRACE_PTR (ExConvertToString, ObjDesc);


454     switch (ObjDesc->Common.Type)
455     {
456     case ACPI_TYPE_STRING:

458         /* No conversion necessary */

460         *ResultDesc = ObjDesc;
461         return_ACPI_STATUS (AE_OK);


463     case ACPI_TYPE_INTEGER:

465         switch (Type)
466         {
467         case ACPI_EXPLICIT_CONVERT_DECIMAL:

469             /* Make room for maximum decimal number */

471             StringLength = ACPI_MAX_DECIMAL_DIGITS;
472             Base = 10;
473             break;

475         default:

477             /* Two hex string characters for each integer byte */

479             StringLength = ACPI_MUL_2 (AcpiGbl_IntegerByteWidth);
480             break;
481         }

483         /*
484          * Create a new String
485          * Need enough space for one ASCII integer (plus null terminator)
486          */
487         ReturnDesc = AcpiUtCreateStringObject ((ACPI_SIZE) StringLength);
488         if (!ReturnDesc)
489         {
490             return_ACPI_STATUS (AE_NO_MEMORY);
491         }

493         NewBuf = ReturnDesc->Buffer.Pointer;

495         /* Convert integer to string */

497         StringLength = AcpiExConvertToAscii (ObjDesc->Integer.Value, Base,
498                             NewBuf, AcpiGbl_IntegerByteWidth);

500         /* Null terminate at the correct place */

502         ReturnDesc->String.Length = StringLength;
503         NewBuf [StringLength] = 0;
504         break;


506     case ACPI_TYPE_BUFFER:

508         /* Setup string length, base, and separator */

510         switch (Type)
511         {
```

```
512         case ACPI_EXPLICIT_CONVERT_DECIMAL: /* Used by ToDecimalString */
513             /*
514              * From ACPI: "If Data is a buffer, it is converted to a string of
515              * decimal values separated by commas."
516              */
517             Base = 10;
518
519             /*
520              * Calculate the final string length. Individual string values
521              * are variable length (include separator for each)
522              */
523             for (i = 0; i < ObjDesc->Buffer.Length; i++)
524             {
525                 if (ObjDesc->Buffer.Pointer[i] >= 100)
526                 {
527                     StringLength += 4;
528                 }
529                 else if (ObjDesc->Buffer.Pointer[i] >= 10)
530                 {
531                     StringLength += 3;
532                 }
533                 else
534                 {
535                     StringLength += 2;
536                 }
537             }
538             break;
539
540         case ACPI_IMPLICIT_CONVERT_HEX:
541             /*
542              * From the ACPI spec:
543              *"The entire contents of the buffer are converted to a string of
544              * two-character hexadecimal numbers, each separated by a space."
545              */
546             Separator = ' ';
547             StringLength = (ObjDesc->Buffer.Length * 3);
548             break;
549
550         case ACPI_EXPLICIT_CONVERT_HEX:     /* Used by ToHexString */
551             /*
552              * From ACPI: "If Data is a buffer, it is converted to a string of
553              * hexadecimal values separated by commas."
554              */
555             StringLength = (ObjDesc->Buffer.Length * 3);
556             break;
557
558         default:
559             return_ACPI_STATUS (AE_BAD_PARAMETER);
560         }
561
562         /*
563          * Create a new string object and string buffer
564          * (-1 because of extra separator included in StringLength from above)
565          * Allow creation of zero-length strings from zero-length buffers.
566          */
567         if (StringLength)
568         {
569             StringLength--;
570         }
571
572         ReturnDesc = AcpiUtCreateStringObject ((ACPI_SIZE) StringLength);
573         if (!ReturnDesc)
574         {
575             return_ACPI_STATUS (AE_NO_MEMORY);
576         }
```



```
578         NewBuf = ReturnDesc->Buffer.Pointer;
579
580         /*
581          * Convert buffer bytes to hex or decimal values
582          * (separated by commas or spaces)
583          */
584         for (i = 0; i < ObjDesc->Buffer.Length; i++)
585         {
586             NewBuf += AcpiExConvertToAscii (
587                         (UINT64) ObjDesc->Buffer.Pointer[i], Base,
588                         NewBuf, 1);
589             *NewBuf++ = Separator; /* each separated by a comma or space */
590         }
591
592         /*
593          * Null terminate the string
594          * (overwrites final comma/space from above)
595          */
596         if (ObjDesc->Buffer.Length)
597         {
598             NewBuf--;
599         }
600         *NewBuf = 0;
601         break;
602
603     default:
604
605         return_ACPI_STATUS (AE_TYPE);
606     }
607
608     *ResultDesc = ReturnDesc;
609     return_ACPI_STATUS (AE_OK);
610 }
611
612
613 /*******************************************************************************
614  *
615  * FUNCTION:    AcpiExConvertToTargetType
616  *
617  * PARAMETERS:  DestinationType     - Current type of the destination
618  *              SourceDesc          - Source object to be converted.
619  *              ResultDesc          - Where the converted object is returned
620  *              WalkState           - Current method state
621  *
622  * RETURN:      Status
623  *
624  * DESCRIPTION: Implements "implicit conversion" rules for storing an object.
625  *
626  ******************************************************************************/
627
628 ACPI_STATUS
629 AcpiExConvertToTargetType (
630     ACPI_OBJECT_TYPE        DestinationType,
631     ACPI_OPERAND_OBJECT     *SourceDesc,
632     ACPI_OPERAND_OBJECT     **ResultDesc,
633     ACPI_WALK_STATE         *WalkState)
634 {
635     ACPI_STATUS             Status = AE_OK;
636
637
638     ACPI_FUNCTION_TRACE (ExConvertToTargetType);
639
640
641     /* Default behavior */
642
643     *ResultDesc = SourceDesc;
```

```
645     /*
646      * If required by the target,
647      * perform implicit conversion on the source before we store it.
648      */
649     switch (GET_CURRENT_ARG_TYPE (WalkState->OpInfo->RuntimeArgs))
650     {
651     case ARGI_SIMPLE_TARGET:
652     case ARGI_FIXED_TARGET:
653     case ARGI_INTEGER_REF:      /* Handles Increment, Decrement cases */

655         switch (DestinationType)
656         {
657         case ACPI_TYPE_LOCAL_REGION_FIELD:
658             /*
659              * Named field can always handle conversions
660              */
661             break;

663         default:

665             /* No conversion allowed for these types */

667             if (DestinationType != SourceDesc->Common.Type)
668             {
669                 ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
670                     "Explicit operator, will store (%s) over existing type (%s)\
671                     AcpiUtGetObjectTypeName (SourceDesc),
672                     AcpiUtGetTypeName (DestinationType)));
673                 Status = AE_TYPE;
674             }
675         }
676         break;


678     case ARGI_TARGETREF:

680         switch (DestinationType)
681         {
682         case ACPI_TYPE_INTEGER:
683         case ACPI_TYPE_BUFFER_FIELD:
684         case ACPI_TYPE_LOCAL_BANK_FIELD:
685         case ACPI_TYPE_LOCAL_INDEX_FIELD:
686             /*
687              * These types require an Integer operand. We can convert
688              * a Buffer or a String to an Integer if necessary.
689              */
690             Status = AcpiExConvertToInteger (SourceDesc, ResultDesc,
691                         16);
692             break;


694         case ACPI_TYPE_STRING:
695             /*
696              * The operand must be a String. We can convert an
697              * Integer or Buffer if necessary
698              */
699             Status = AcpiExConvertToString (SourceDesc, ResultDesc,
700                         ACPI_IMPLICIT_CONVERT_HEX);
701             break;


703         case ACPI_TYPE_BUFFER:
704             /*
705              * The operand must be a Buffer. We can convert an
706              * Integer or String if necessary
```



```
707              */
708             Status = AcpiExConvertToBuffer (SourceDesc, ResultDesc);
709             break;

711         default:

716         default:
713             ACPI_ERROR ((AE_INFO, "Bad destination type during conversion: 0x%X"
714                 DestinationType));
715             Status = AE_AML_INTERNAL;
716             break;
717         }
718         break;


720     case ARGI_REFERENCE:
721         /*
722          * CreateXxxxField cases - we are storing the field object into the name
723          */
724         break;

726     default:

732     default:
728         ACPI_ERROR ((AE_INFO,
729             "Unknown Target type ID 0x%X AmlOpcode 0x%X DestType %s",
730             GET_CURRENT_ARG_TYPE (WalkState->OpInfo->RuntimeArgs),
731             WalkState->Opcode, AcpiUtGetTypeName (DestinationType)));
732         Status = AE_AML_INTERNAL;
733     }

735     /*
736      * Source-to-Target conversion semantics:
737      *
738      * If conversion to the target type cannot be performed, then simply
739      * overwrite the target with the new object and type.
740      */
741     if (Status == AE_TYPE)
742     {
743         Status = AE_OK;
744     }

746     return_ACPI_STATUS (Status);
747 }
```

****
16473 Thu Dec 26 13:49:02 2013
new/usr/src/common/acpica/components/executer/excreate.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: excreate - Named object creation
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXCREATE_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acinterp.h"
  49 #include "amlcode.h"
  50 #include "acnamesp.h"


  53 #define _COMPONENT          ACPI_EXECUTER
  54         ACPI_MODULE_NAME    ("excreate")


  57 #ifndef ACPI_NO_METHOD_EXECUTION
  58 /*****************************************************************************
  59  *
```



```
  60  * FUNCTION:    AcpiExCreateAlias
  61  *
  62  * PARAMETERS:  WalkState            - Current state, contains operands
  63  *
  64  * RETURN:      Status
  65  *
  66  * DESCRIPTION: Create a new named alias
  67  *
  68  *****************************************************************************/

  70 ACPI_STATUS
  71 AcpiExCreateAlias (
  72     ACPI_WALK_STATE         *WalkState)
  73 {
  74     ACPI_NAMESPACE_NODE     *TargetNode;
  75     ACPI_NAMESPACE_NODE     *AliasNode;
  76     ACPI_STATUS             Status = AE_OK;


  79     ACPI_FUNCTION_TRACE (ExCreateAlias);


  82     /* Get the source/alias operands (both namespace nodes) */

  84     AliasNode =  (ACPI_NAMESPACE_NODE *) WalkState->Operands[0];
  85     TargetNode = (ACPI_NAMESPACE_NODE *) WalkState->Operands[1];

  87     if ((TargetNode->Type == ACPI_TYPE_LOCAL_ALIAS)  ||
  88         (TargetNode->Type == ACPI_TYPE_LOCAL_METHOD_ALIAS))
  89     {
  90         /*
  91          * Dereference an existing alias so that we don't create a chain
  92          * of aliases. With this code, we guarantee that an alias is
  93          * always exactly one level of indirection away from the
  94          * actual aliased name.
  95          */
  96         TargetNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, TargetNode->Object);
  97     }

  99     /*
 100      * For objects that can never change (i.e., the NS node will
 101      * permanently point to the same object), we can simply attach
 102      * the object to the new NS node. For other objects (such as
 103      * Integers, buffers, etc.), we have to point the Alias node
 104      * to the original Node.
 105      */
 106     switch (TargetNode->Type)
 107     {

 109     /* For these types, the sub-object can change dynamically via a Store */

 111     case ACPI_TYPE_INTEGER:
 112     case ACPI_TYPE_STRING:
 113     case ACPI_TYPE_BUFFER:
 114     case ACPI_TYPE_PACKAGE:
 115     case ACPI_TYPE_BUFFER_FIELD:

 116     /*
 117      * These types open a new scope, so we need the NS node in order to access
 118      * any children.
 119      */
 120     case ACPI_TYPE_DEVICE:
 121     case ACPI_TYPE_POWER:
 122     case ACPI_TYPE_PROCESSOR:
 123     case ACPI_TYPE_THERMAL:
 124     case ACPI_TYPE_LOCAL_SCOPE:
```

```
125         /*
126          * The new alias has the type ALIAS and points to the original
127          * NS node, not the object itself.
128          */
129         AliasNode->Type = ACPI_TYPE_LOCAL_ALIAS;
130         AliasNode->Object = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, TargetNode);
131         break;

133     case ACPI_TYPE_METHOD:

134         /*
135          * Control method aliases need to be differentiated
136          */
137         AliasNode->Type = ACPI_TYPE_LOCAL_METHOD_ALIAS;
138         AliasNode->Object = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, TargetNode);
139         break;

141     default:

143         /* Attach the original source object to the new Alias Node */

145         /*
146          * The new alias assumes the type of the target, and it points
147          * to the same object. The reference count of the object has an
148          * additional reference to prevent deletion out from under either the
149          * target node or the alias Node
150          */
151         Status = AcpiNsAttachObject (AliasNode,
152                     AcpiNsGetAttachedObject (TargetNode), TargetNode->Type);
153         break;
154     }

156     /* Since both operands are Nodes, we don't need to delete them */

158     return_ACPI_STATUS (Status);
159 }
```

unchanged_portion_omitted_

```
278 /*****************************************************************************
279  *
280  * FUNCTION:    AcpiExCreateRegion
281  *
282  * PARAMETERS:  AmlStart            - Pointer to the region declaration AML
283  *              AmlLength           - Max length of the declaration AML
284  *              SpaceId             - Address space ID for the region
287  *              RegionSpace         - SpaceID for the region
285  *              WalkState           - Current state
286  *
287  * RETURN:      Status
288  *
289  * DESCRIPTION: Create a new operation region object
290  *
291  ****************************************************************************/

293 ACPI_STATUS
294 AcpiExCreateRegion (
295     UINT8                   *AmlStart,
296     UINT32                  AmlLength,
297     UINT8                   SpaceId,
300     UINT8                   RegionSpace,
298     ACPI_WALK_STATE         *WalkState)
299 {
300     ACPI_STATUS             Status;
301     ACPI_OPERAND_OBJECT     *ObjDesc;
```



```
302     ACPI_NAMESPACE_NODE     *Node;
303     ACPI_OPERAND_OBJECT     *RegionObj2;


306     ACPI_FUNCTION_TRACE (ExCreateRegion);


309     /* Get the Namespace Node */

311     Node = WalkState->Op->Common.Node;

313     /*
314      * If the region object is already attached to this node,
315      * just return
316      */
317     if (AcpiNsGetAttachedObject (Node))
318     {
319         return_ACPI_STATUS (AE_OK);
320     }

322     /*
323      * Space ID must be one of the predefined IDs, or in the user-defined
324      * range
325      */
326     if (!AcpiIsValidSpaceId (SpaceId))
329     if ((RegionSpace >= ACPI_NUM_PREDEFINED_REGIONS) &&
330         (RegionSpace < ACPI_USER_REGION_BEGIN) &&
331         (RegionSpace != ACPI_ADR_SPACE_DATA_TABLE))
327     {
328         /*
329          * Print an error message, but continue. We don't want to abort
330          * a table load for this exception. Instead, if the region is
331          * actually used at runtime, abort the executing method.
332          */
333         ACPI_ERROR ((AE_INFO, "Invalid/unknown Address Space ID: 0x%2.2X", Space
333         ACPI_ERROR ((AE_INFO, "Invalid AddressSpace type 0x%X", RegionSpace));
334         return_ACPI_STATUS (AE_AML_INVALID_SPACE_ID);
334     }

336     ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "Region Type - %s (0x%X)\n",
337         AcpiUtGetRegionName (SpaceId), SpaceId));
338         AcpiUtGetRegionName (RegionSpace), RegionSpace));

339     /* Create the region descriptor */

341     ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_REGION);
342     if (!ObjDesc)
343     {
344         Status = AE_NO_MEMORY;
345         goto Cleanup;
346     }

348     /*
349      * Remember location in AML stream of address & length
350      * operands since they need to be evaluated at run time.
351      */
352     RegionObj2 = ObjDesc->Common.NextObject;
353     RegionObj2->Extra.AmlStart = AmlStart;
354     RegionObj2->Extra.AmlLength = AmlLength;
355     if (WalkState->ScopeInfo)
356     {
357         RegionObj2->Extra.ScopeNode = WalkState->ScopeInfo->Scope.Node;
358     }
359     else
360     {
361         RegionObj2->Extra.ScopeNode = Node;
```

```
362     }

364     /* Init the region from the operands */

366     ObjDesc->Region.SpaceId = SpaceId;
359     ObjDesc->Region.SpaceId = RegionSpace;
367     ObjDesc->Region.Address = 0;
368     ObjDesc->Region.Length = 0;
369     ObjDesc->Region.Node = Node;

371     /* Install the new region object in the parent Node */

373     Status = AcpiNsAttachObject (Node, ObjDesc, ACPI_TYPE_REGION);


376 Cleanup:

378     /* Remove local reference to the object */

380     AcpiUtRemoveReference (ObjDesc);
381     return_ACPI_STATUS (Status);
382 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    8374 Thu Dec 26 13:49:02 2013
new/usr/src/common/acpica/components/executer/exdebug.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exdebug - Support for stores to the AML Debug Object
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXDEBUG_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exdebug")


#ifndef ACPI_NO_ERROR_MESSAGES
/******************************************************************************
 *
 * FUNCTION:    AcpiExDoDebugObject
 *
```



```
 * PARAMETERS:  SourceDesc          - Object to be output to "Debug Object"
 *              Level               - Indentation level (used for packages)
 *              Index               - Current package element, zero if not pkg
 *
 * RETURN:      None
 *
 * DESCRIPTION: Handles stores to the AML Debug Object. For example:
 *              Store(INT1, Debug)
 *
 * This function is not compiled if ACPI_NO_ERROR_MESSAGES is set.
 *
 * This function is only enabled if AcpiGbl_EnableAmlDebugObject is set, or
 * if ACPI_LV_DEBUG_OBJECT is set in the AcpiDbgLevel. Thus, in the normal
 * operational case, stores to the debug object are ignored but can be easily
 * enabled if necessary.
 *
 *****************************************************************************/

void
AcpiExDoDebugObject (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    UINT32                  Level,
    UINT32                  Index)
{
    UINT32                  i;


    ACPI_FUNCTION_TRACE_PTR (ExDoDebugObject, SourceDesc);


    /* Output must be enabled via the DebugObject global or the DbgLevel */

    if (!AcpiGbl_EnableAmlDebugObject &&
        !(AcpiDbgLevel & ACPI_LV_DEBUG_OBJECT))
    {
        return_VOID;
    }

    /*
     * Print line header as long as we are not in the middle of an
     * object display
     */
    if (!((Level > 0) && Index == 0))
    {
        AcpiOsPrintf ("[ACPI Debug] %*s", Level, " ");
    }

    /* Display the index for package output only */

    if (Index > 0)
    {
       AcpiOsPrintf ("(%.2u) ", Index-1);
    }

    if (!SourceDesc)
    {
        AcpiOsPrintf ("[Null Object]\n");
        return_VOID;
    }

    if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc) == ACPI_DESC_TYPE_OPERAND)
    {
        AcpiOsPrintf ("%s ", AcpiUtGetObjectTypeName (SourceDesc));

        if (!AcpiUtValidInternalObject (SourceDesc))
        {
```

```
126             AcpiOsPrintf ("%p, Invalid Internal Object!\n", SourceDesc);
127             return_VOID;
128         }
129     }
130     else if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc) == ACPI_DESC_TYPE_NAMED)
131     {
132         AcpiOsPrintf ("%s: %p\n",
133             AcpiUtGetTypeName (((ACPI_NAMESPACE_NODE *) SourceDesc)->Type),
134             SourceDesc);
135         return_VOID;
136     }
137     else
138     {
139         return_VOID;
140     }
142     /* SourceDesc is of type ACPI_DESC_TYPE_OPERAND */
144     switch (SourceDesc->Common.Type)
145     {
146     case ACPI_TYPE_INTEGER:
148         /* Output correct integer width */
150         if (AcpiGbl_IntegerByteWidth == 4)
151         {
152             AcpiOsPrintf ("0x%8.8X\n",
153                 (UINT32) SourceDesc->Integer.Value);
154         }
155         else
156         {
157             AcpiOsPrintf ("0x%8.8X%8.8X\n",
158                 ACPI_FORMAT_UINT64 (SourceDesc->Integer.Value));
159         }
160         break;
162     case ACPI_TYPE_BUFFER:
164         AcpiOsPrintf ("[0x%.2X]\n", (UINT32) SourceDesc->Buffer.Length);
165         AcpiUtDumpBuffer (SourceDesc->Buffer.Pointer,
165         AcpiUtDumpBuffer2 (SourceDesc->Buffer.Pointer,
166             (SourceDesc->Buffer.Length < 256) ?
167                 SourceDesc->Buffer.Length : 256, DB_BYTE_DISPLAY, 0);
167                 SourceDesc->Buffer.Length : 256, DB_BYTE_DISPLAY);
168         break;
170     case ACPI_TYPE_STRING:
172         AcpiOsPrintf ("[0x%.2X] \"%s\"\n",
173             SourceDesc->String.Length, SourceDesc->String.Pointer);
174         break;
176     case ACPI_TYPE_PACKAGE:
178         AcpiOsPrintf ("[Contains 0x%.2X Elements]\n",
179             SourceDesc->Package.Count);
181         /* Output the entire contents of the package */
183         for (i = 0; i < SourceDesc->Package.Count; i++)
184         {
185             AcpiExDoDebugObject (SourceDesc->Package.Elements[i],
186                 Level+4, i+1);
187         }
188         break;
```



```
190     case ACPI_TYPE_LOCAL_REFERENCE:
192         AcpiOsPrintf ("[%s] ", AcpiUtGetReferenceName (SourceDesc));
194         /* Decode the reference */
196         switch (SourceDesc->Reference.Class)
197         {
198         case ACPI_REFCLASS_INDEX:
200             AcpiOsPrintf ("0x%X\n", SourceDesc->Reference.Value);
201             break;
203         case ACPI_REFCLASS_TABLE:
205             /* Case for DdbHandle */
207             AcpiOsPrintf ("Table Index 0x%X\n", SourceDesc->Reference.Value);
208             return_VOID;
208             return;
210         default:
212             break;
213         }
215         AcpiOsPrintf ("  ");
217         /* Check for valid node first, then valid object */
219         if (SourceDesc->Reference.Node)
220         {
221             if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc->Reference.Node) !=
222                     ACPI_DESC_TYPE_NAMED)
223             {
224                 AcpiOsPrintf (" %p - Not a valid namespace node\n",
225                     SourceDesc->Reference.Node);
226             }
227             else
228             {
229                 AcpiOsPrintf ("Node %p [%4.4s] ", SourceDesc->Reference.Node,
230                     (SourceDesc->Reference.Node)->Name.Ascii);
232                 switch ((SourceDesc->Reference.Node)->Type)
233                 {
234                 /* These types have no attached object */
236                 case ACPI_TYPE_DEVICE:
237                     AcpiOsPrintf ("Device\n");
238                     break;
240                 case ACPI_TYPE_THERMAL:
241                     AcpiOsPrintf ("Thermal Zone\n");
242                     break;
244                 default:
246                     AcpiExDoDebugObject ((SourceDesc->Reference.Node)->Object,
247                         Level+4, 0);
248                     break;
249                 }
250             }
251         }
252         else if (SourceDesc->Reference.Object)
253         {
254             if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc->Reference.Object) ==
```

```
255                     ACPI_DESC_TYPE_NAMED)
256             {
257                 AcpiExDoDebugObject (((ACPI_NAMESPACE_NODE *)
258                     SourceDesc->Reference.Object)->Object,
259                     Level+4, 0);
260             }
261             else
262             {
263                 AcpiExDoDebugObject (SourceDesc->Reference.Object,
264                     Level+4, 0);
265             }
266         }
267         break;

269     default:

271         AcpiOsPrintf ("%p\n", SourceDesc);
272         break;
273     }

275     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_EXEC, "\n"));
276     return_VOID;
277 }
278 #endif
```

```
**********************************************************
   34334 Thu Dec 26 13:49:02 2013
new/usr/src/common/acpica/components/executer/exdump.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exdump - Interpreter debug output routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXDUMP_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exdump")

/*
 * The following routines are used for debug output only
 */
#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)
```



```
/* Local prototypes */

static void
AcpiExOutString (
    char                    *Title,
    char                    *Value);

static void
AcpiExOutPointer (
    char                    *Title,
    void                    *Value);

static void
AcpiExDumpObject (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_EXDUMP_INFO        *Info);

static void
AcpiExDumpReferenceObj (
    ACPI_OPERAND_OBJECT     *ObjDesc);

static void
AcpiExDumpPackageObj (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  Level,
    UINT32                  Index);


/******************************************************************************
 *
 * Object Descriptor info tables
 *
 * Note: The first table entry must be an INIT opcode and must contain
 * the table length (number of table entries)
 *
 *****************************************************************************/

static ACPI_EXDUMP_INFO     AcpiExDumpInteger[2] =
{
    {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpInteger),        NULL},
    {ACPI_EXD_UINT64,   ACPI_EXD_OFFSET (Integer.Value),                "Value"}
};
```

_____***unchanged_portion_omitted_***

```
static ACPI_EXDUMP_INFO     AcpiExDumpDevice[4] =
{
    {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpDevice),         NULL},
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Device.Handler),               "Handler
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Device.NotifyList[0]),         "System
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Device.NotifyList[1]),         "Device
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Device.SystemNotify),          "System
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Device.DeviceNotify),          "Device
};
```

_____***unchanged_portion_omitted_***

```
static ACPI_EXDUMP_INFO     AcpiExDumpPower[5] =
{
    {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpPower),          NULL},
    {ACPI_EXD_UINT32,   ACPI_EXD_OFFSET (PowerResource.SystemLevel),    "System
    {ACPI_EXD_UINT32,   ACPI_EXD_OFFSET (PowerResource.ResourceOrder),  "Resourc
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (PowerResource.NotifyList[0]),  "System
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (PowerResource.NotifyList[1]),  "Device
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (PowerResource.SystemNotify),   "System
    {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (PowerResource.DeviceNotify),   "Device
};
```

```
186 static ACPI_EXDUMP_INFO     AcpiExDumpProcessor[7] =
187 {
188     {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpProcessor),      NULL},
189     {ACPI_EXD_UINT8,    ACPI_EXD_OFFSET (Processor.ProcId),             "Process
190     {ACPI_EXD_UINT8 ,   ACPI_EXD_OFFSET (Processor.Length),             "Length"
191     {ACPI_EXD_ADDRESS,  ACPI_EXD_OFFSET (Processor.Address),            "Address
192     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Processor.NotifyList[0]),      "System
193     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Processor.NotifyList[1]),      "Device
192     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Processor.SystemNotify),       "System
193     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Processor.DeviceNotify),       "Device
194     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Processor.Handler),            "Handler
195 };

197 static ACPI_EXDUMP_INFO     AcpiExDumpThermal[4] =
198 {
199     {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpThermal),        NULL},
200     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (ThermalZone.NotifyList[0]),    "System
201     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (ThermalZone.NotifyList[1]),    "Device
200     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (ThermalZone.SystemNotify),     "System
201     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (ThermalZone.DeviceNotify),     "Device
202     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (ThermalZone.Handler),          "Handler
203 };
______unchanged_portion_omitted_

212 static ACPI_EXDUMP_INFO     AcpiExDumpRegionField[5] =
212 static ACPI_EXDUMP_INFO     AcpiExDumpRegionField[3] =
213 {
214     {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpRegionField),    NULL},
215     {ACPI_EXD_FIELD,    0,                                              NULL},
216     {ACPI_EXD_UINT8,    ACPI_EXD_OFFSET (Field.AccessLength),           "AccessL
217     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Field.RegionObj),              "Region
218     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Field.ResourceBuffer),         "Resourc
216     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Field.RegionObj),              "Region
219 };
______unchanged_portion_omitted_

261 static ACPI_EXDUMP_INFO     AcpiExDumpNotify[7] =
259 static ACPI_EXDUMP_INFO     AcpiExDumpNotify[3] =
262 {
263     {ACPI_EXD_INIT,     ACPI_EXD_TABLE_SIZE (AcpiExDumpNotify),         NULL},
264     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Node),                  "Node"},
265     {ACPI_EXD_UINT32,   ACPI_EXD_OFFSET (Notify.HandlerType),           "Handler
266     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Handler),               "Handler
267     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Context),               "Context
268     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Next[0]),               "Next Sy
269     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Next[1]),               "Next De
263     {ACPI_EXD_POINTER,  ACPI_EXD_OFFSET (Notify.Context),               "Context
270 };
______unchanged_portion_omitted_


339 /*******************************************************************************
340  *
341  * FUNCTION:    AcpiExDumpObject
342  *
343  * PARAMETERS:  ObjDesc             - Descriptor to dump
344  *              Info                - Info table corresponding to this object
345  *                                    type
346  *
347  * RETURN:      None
348  *
349  * DESCRIPTION: Walk the info table for this object
350  *
351  ******************************************************************************/
```



```
353 static void
354 AcpiExDumpObject (
355     ACPI_OPERAND_OBJECT     *ObjDesc,
356     ACPI_EXDUMP_INFO        *Info)
357 {
358     UINT8                   *Target;
359     char                    *Name;
360     const char              *ReferenceName;
361     UINT8                   Count;


364     if (!Info)
365     {
366         AcpiOsPrintf (
367             "ExDumpObject: Display not implemented for object type %s\n",
368             AcpiUtGetObjectTypeName (ObjDesc));
369         return;
370     }

372     /* First table entry must contain the table length (# of table entries) */

374     Count = Info->Offset;

376     while (Count)
377     {
378         Target = ACPI_ADD_PTR (UINT8, ObjDesc, Info->Offset);
379         Name = Info->Name;

381         switch (Info->Opcode)
382         {
383         case ACPI_EXD_INIT:

385             break;

387         case ACPI_EXD_TYPE:

389             AcpiExOutString  ("Type", AcpiUtGetObjectTypeName (ObjDesc));
390             break;

392         case ACPI_EXD_UINT8:

394             AcpiOsPrintf ("%20s : %2.2X\n", Name, *Target);
395             break;

397         case ACPI_EXD_UINT16:

399             AcpiOsPrintf ("%20s : %4.4X\n", Name, ACPI_GET16 (Target));
400             break;

402         case ACPI_EXD_UINT32:

404             AcpiOsPrintf ("%20s : %8.8X\n", Name, ACPI_GET32 (Target));
405             break;

407         case ACPI_EXD_UINT64:

409             AcpiOsPrintf ("%20s : %8.8X%8.8X\n", "Value",
410                 ACPI_FORMAT_UINT64 (ACPI_GET64 (Target)));
411             break;

413         case ACPI_EXD_POINTER:
414         case ACPI_EXD_ADDRESS:

416             AcpiExOutPointer (Name, *ACPI_CAST_PTR (void *, Target));
417             break;
```

```
419         case ACPI_EXD_STRING:

421             AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
422             AcpiOsPrintf ("\n");
423             break;

425         case ACPI_EXD_BUFFER:

427             ACPI_DUMP_BUFFER (ObjDesc->Buffer.Pointer, ObjDesc->Buffer.Length);
428             break;

430         case ACPI_EXD_PACKAGE:

432             /* Dump the package contents */

434             AcpiOsPrintf ("\nPackage Contents:\n");
435             AcpiExDumpPackageObj (ObjDesc, 0, 0);
436             break;

438         case ACPI_EXD_FIELD:

440             AcpiExDumpObject (ObjDesc, AcpiExDumpFieldCommon);
441             break;

443         case ACPI_EXD_REFERENCE:

445             ReferenceName = AcpiUtGetReferenceName (ObjDesc);
446             AcpiExOutString ("Class Name", ACPI_CAST_PTR (char, ReferenceName));
437             AcpiExOutString ("Class Name",
438                 ACPI_CAST_PTR (char, AcpiUtGetReferenceName (ObjDesc)));
447             AcpiExDumpReferenceObj (ObjDesc);
448             break;

450         default:

452             AcpiOsPrintf ("**** Invalid table opcode [%X] ****\n",
453                 Info->Opcode);
454             return;
455         }

457         Info++;
458         Count--;
459     }
460 }


463 /*****************************************************************************
464  *
465  * FUNCTION:    AcpiExDumpOperand
466  *
467  * PARAMETERS:  *ObjDesc        - Pointer to entry to be dumped
468  *              Depth           - Current nesting depth
469  *
470  * RETURN:      None
471  *
472  * DESCRIPTION: Dump an operand object
473  *
474  ****************************************************************************/

476 void
477 AcpiExDumpOperand (
478     ACPI_OPERAND_OBJECT     *ObjDesc,
479     UINT32                  Depth)
480 {
481     UINT32                  Length;
482     UINT32                  Index;
```



```
485     ACPI_FUNCTION_NAME (ExDumpOperand)


488     /* Check if debug output enabled */

490     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_EXEC, _COMPONENT))
480     if (!((ACPI_LV_EXEC & AcpiDbgLevel) && (_COMPONENT & AcpiDbgLayer)))
491     {
492         return;
493     }

495     if (!ObjDesc)
496     {
497         /* This could be a null element of a package */

499         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Null Object Descriptor\n"));
500         return;
501     }

503     if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) == ACPI_DESC_TYPE_NAMED)
504     {
505         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "%p Namespace Node: ", ObjDesc));
506         ACPI_DUMP_ENTRY (ObjDesc, ACPI_LV_EXEC);
507         return;
508     }

510     if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND)
511     {
512         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
513             "%p is not a node or operand object: [%s]\n",
514             ObjDesc, AcpiUtGetDescriptorName (ObjDesc)));
515         ACPI_DUMP_BUFFER (ObjDesc, sizeof (ACPI_OPERAND_OBJECT));
516         return;
517     }

519     /* ObjDesc is a valid object */

521     if (Depth > 0)
522     {
523         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "%*s[%u] %p ",
524             Depth, " ", Depth, ObjDesc));
525     }
526     else
527     {
528         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "%p ", ObjDesc));
529     }

531     /* Decode object type */

533     switch (ObjDesc->Common.Type)
534     {
535     case ACPI_TYPE_LOCAL_REFERENCE:

537         AcpiOsPrintf ("Reference: [%s] ", AcpiUtGetReferenceName (ObjDesc));

539         switch (ObjDesc->Reference.Class)
540         {
541         case ACPI_REFCLASS_DEBUG:

543             AcpiOsPrintf ("\n");
544             break;


546         case ACPI_REFCLASS_INDEX:
```

```
548             AcpiOsPrintf ("%p\n", ObjDesc->Reference.Object);
549             break;


551         case ACPI_REFCLASS_TABLE:

553             AcpiOsPrintf ("Table Index %X\n", ObjDesc->Reference.Value);
554             break;


556         case ACPI_REFCLASS_REFOF:

558             AcpiOsPrintf ("%p [%s]\n", ObjDesc->Reference.Object,
559                 AcpiUtGetTypeName (((ACPI_OPERAND_OBJECT *)
560                     ObjDesc->Reference.Object)->Common.Type));
561             break;


563         case ACPI_REFCLASS_NAME:

565             AcpiOsPrintf ("- [%4.4s]\n", ObjDesc->Reference.Node->Name.Ascii);
566             break;


568         case ACPI_REFCLASS_ARG:
569         case ACPI_REFCLASS_LOCAL:

571             AcpiOsPrintf ("%X\n", ObjDesc->Reference.Value);
572             break;


574         default:    /* Unknown reference class */

576             AcpiOsPrintf ("%2.2X\n", ObjDesc->Reference.Class);
577             break;
578         }
579         break;


581     case ACPI_TYPE_BUFFER:

583         AcpiOsPrintf ("Buffer length %.2X @ %p\n",
584             ObjDesc->Buffer.Length, ObjDesc->Buffer.Pointer);

586         /* Debug only -- dump the buffer contents */

588         if (ObjDesc->Buffer.Pointer)
589         {
590             Length = ObjDesc->Buffer.Length;
591             if (Length > 128)
592             {
593                 Length = 128;
594             }

596             AcpiOsPrintf ("Buffer Contents: (displaying length 0x%.2X)\n",
597                 Length);
598             ACPI_DUMP_BUFFER (ObjDesc->Buffer.Pointer, Length);
599         }
600         break;


602     case ACPI_TYPE_INTEGER:

604         AcpiOsPrintf ("Integer %8.8X%8.8X\n",
605             ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
```



```
606         break;


608     case ACPI_TYPE_PACKAGE:

610         AcpiOsPrintf ("Package [Len %X] ElementArray %p\n",
611             ObjDesc->Package.Count, ObjDesc->Package.Elements);

613         /*
614          * If elements exist, package element pointer is valid,
615          * and debug_level exceeds 1, dump package's elements.
616          */
617         if (ObjDesc->Package.Count &&
618             ObjDesc->Package.Elements &&
619             AcpiDbgLevel > 1)
620         {
621             for (Index = 0; Index < ObjDesc->Package.Count; Index++)
622             {
623                 AcpiExDumpOperand (ObjDesc->Package.Elements[Index], Depth+1);
624             }
625         }
626         break;


628     case ACPI_TYPE_REGION:

630         AcpiOsPrintf ("Region %s (%X)",
631             AcpiUtGetRegionName (ObjDesc->Region.SpaceId),
632             ObjDesc->Region.SpaceId);

634         /*
635          * If the address and length have not been evaluated,
636          * don't print them.
637          */
638         if (!(ObjDesc->Region.Flags & AOPOBJ_DATA_VALID))
639         {
640             AcpiOsPrintf ("\n");
641         }
642         else
643         {
644             AcpiOsPrintf (" base %8.8X%8.8X Length %X\n",
645                 ACPI_FORMAT_NATIVE_UINT (ObjDesc->Region.Address),
646                 ObjDesc->Region.Length);
647         }
648         break;


650     case ACPI_TYPE_STRING:

652         AcpiOsPrintf ("String length %X @ %p ",
653             ObjDesc->String.Length,
654             ObjDesc->String.Pointer);

656         AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
657         AcpiOsPrintf ("\n");
658         break;


660     case ACPI_TYPE_LOCAL_BANK_FIELD:

662         AcpiOsPrintf ("BankField\n");
663         break;


665     case ACPI_TYPE_LOCAL_REGION_FIELD:
```

```
667         AcpiOsPrintf ("RegionField: Bits=%X AccWidth=%X Lock=%X Update=%X at "
668             "byte=%X bit=%X of below:\n",
669             ObjDesc->Field.BitLength,
670             ObjDesc->Field.AccessByteWidth,
671             ObjDesc->Field.FieldFlags & AML_FIELD_LOCK_RULE_MASK,
672             ObjDesc->Field.FieldFlags & AML_FIELD_UPDATE_RULE_MASK,
673             ObjDesc->Field.BaseByteOffset,
674             ObjDesc->Field.StartFieldBitOffset);

676         AcpiExDumpOperand (ObjDesc->Field.RegionObj, Depth+1);
677         break;


679     case ACPI_TYPE_LOCAL_INDEX_FIELD:

681         AcpiOsPrintf ("IndexField\n");
682         break;


684     case ACPI_TYPE_BUFFER_FIELD:

686         AcpiOsPrintf ("BufferField: %X bits at byte %X bit %X of\n",
687             ObjDesc->BufferField.BitLength,
688             ObjDesc->BufferField.BaseByteOffset,
689             ObjDesc->BufferField.StartFieldBitOffset);

691         if (!ObjDesc->BufferField.BufferObj)
692         {
693             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "*NULL*\n"));
694         }
695         else if ((ObjDesc->BufferField.BufferObj)->Common.Type !=
696                     ACPI_TYPE_BUFFER)
697         {
698             AcpiOsPrintf ("*not a Buffer*\n");
699         }
700         else
701         {
702             AcpiExDumpOperand (ObjDesc->BufferField.BufferObj, Depth+1);
703         }
704         break;


706     case ACPI_TYPE_EVENT:

708         AcpiOsPrintf ("Event\n");
709         break;


711     case ACPI_TYPE_METHOD:

713         AcpiOsPrintf ("Method(%X) @ %p:%X\n",
714             ObjDesc->Method.ParamCount,
715             ObjDesc->Method.AmlStart,
716             ObjDesc->Method.AmlLength);
717         break;


719     case ACPI_TYPE_MUTEX:

721         AcpiOsPrintf ("Mutex\n");
722         break;


724     case ACPI_TYPE_DEVICE:

726         AcpiOsPrintf ("Device\n");
```



```
727         break;


729     case ACPI_TYPE_POWER:

731         AcpiOsPrintf ("Power\n");
732         break;


734     case ACPI_TYPE_PROCESSOR:

736         AcpiOsPrintf ("Processor\n");
737         break;


739     case ACPI_TYPE_THERMAL:

741         AcpiOsPrintf ("Thermal\n");
742         break;

744     default:

757     default:
746         /* Unknown Type */

748         AcpiOsPrintf ("Unknown Type %X\n", ObjDesc->Common.Type);
749         break;
750     }

752     return;
753 }
______unchanged_portion_omitted_


836 /*******************************************************************************
837  *
838  * FUNCTION:    AcpiExDumpNamespaceNode
839  *
840  * PARAMETERS:  Node                - Descriptor to dump
841  *              Flags               - Force display if TRUE
842  *
843  * DESCRIPTION: Dumps the members of the given.Node
844  *
845  ******************************************************************************/

847 void
848 AcpiExDumpNamespaceNode (
849     ACPI_NAMESPACE_NODE     *Node,
850     UINT32                  Flags)
851 {

853     ACPI_FUNCTION_ENTRY ();


856     if (!Flags)
857     {
858         /* Check if debug output enabled */

860         if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_OBJECTS, _COMPONENT))
870         if (!((ACPI_LV_OBJECTS & AcpiDbgLevel) && (_COMPONENT & AcpiDbgLayer)))
861         {
862             return;
863         }
864     }

866     AcpiOsPrintf ("%20s : %4.4s\n", "Name", AcpiUtGetNodeName (Node));
```

```
 867     AcpiExOutString  ("Type", AcpiUtGetTypeName (Node->Type));
 868     AcpiExOutPointer ("Attached Object", AcpiNsGetAttachedObject (Node));
 869     AcpiExOutPointer ("Parent", Node->Parent);

 871     AcpiExDumpObject (ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Node),
 872         AcpiExDumpNode);
 873 }
_____unchanged_portion_omitted_


 935 /*****************************************************************************
 936  *
 937  * FUNCTION:    AcpiExDumpPackageObj
 938  *
 939  * PARAMETERS:  ObjDesc             - Descriptor to dump
 940  *              Level               - Indentation Level
 941  *              Index               - Package index for this object
 942  *
 943  * DESCRIPTION: Dumps the elements of the package
 944  *
 945  ****************************************************************************/

 947 static void
 948 AcpiExDumpPackageObj (
 949     ACPI_OPERAND_OBJECT     *ObjDesc,
 950     UINT32                  Level,
 951     UINT32                  Index)
 952 {
 953     UINT32                  i;


 956     /* Indentation and index output */

 958     if (Level > 0)
 959     {
 960         for (i = 0; i < Level; i++)
 961         {
 962             AcpiOsPrintf ("  ");
 963         }

 965         AcpiOsPrintf ("[%.2d] ", Index);
 966     }

 968     AcpiOsPrintf ("%p ", ObjDesc);

 970     /* Null package elements are allowed */

 972     if (!ObjDesc)
 973     {
 974         AcpiOsPrintf ("[Null Object]\n");
 975         return;
 976     }

 978     /* Packages may only contain a few object types */

 980     switch (ObjDesc->Common.Type)
 981     {
 982     case ACPI_TYPE_INTEGER:

 984         AcpiOsPrintf ("[Integer] = %8.8X%8.8X\n",
 985             ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
 986         break;


 988     case ACPI_TYPE_STRING:
```



```
 990         AcpiOsPrintf ("[String]  Value: ");
 991         AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
1002         for (i = 0; i < ObjDesc->String.Length; i++)
1003         {
1004             AcpiOsPrintf ("%c", ObjDesc->String.Pointer[i]);
1005         }
 992         AcpiOsPrintf ("\n");
 993         break;


 995     case ACPI_TYPE_BUFFER:

 997         AcpiOsPrintf ("[Buffer] Length %.2X = ", ObjDesc->Buffer.Length);
 998         if (ObjDesc->Buffer.Length)
 999         {
1000             AcpiUtDebugDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer
1015             AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
1001                 ObjDesc->Buffer.Length, DB_DWORD_DISPLAY, _COMPONENT);
1002         }
1003         else
1004         {
1005             AcpiOsPrintf ("\n");
1006         }
1007         break;


1009     case ACPI_TYPE_PACKAGE:

1011         AcpiOsPrintf ("[Package] Contains %u Elements:\n",
1012             ObjDesc->Package.Count);

1014         for (i = 0; i < ObjDesc->Package.Count; i++)
1015         {
1016             AcpiExDumpPackageObj (ObjDesc->Package.Elements[i], Level+1, i);
1017         }
1018         break;


1020     case ACPI_TYPE_LOCAL_REFERENCE:

1022         AcpiOsPrintf ("[Object Reference] Type [%s] %2.2X",
1023             AcpiUtGetReferenceName (ObjDesc),
1024             ObjDesc->Reference.Class);
1025         AcpiExDumpReferenceObj (ObjDesc);
1026         break;


1028     default:

1030         AcpiOsPrintf ("[Unknown Type] %X\n", ObjDesc->Common.Type);
1031         break;
1032     }
1033 }


1036 /*****************************************************************************
1037  *
1038  * FUNCTION:    AcpiExDumpObjectDescriptor
1039  *
1040  * PARAMETERS:  ObjDesc             - Descriptor to dump
1041  *              Flags               - Force display if TRUE
1042  *
1043  * DESCRIPTION: Dumps the members of the object descriptor given.
1044  *
1045  ****************************************************************************/
```

```
1047 void
1048 AcpiExDumpObjectDescriptor (
1049     ACPI_OPERAND_OBJECT     *ObjDesc,
1050     UINT32                  Flags)
1051 {
1052     ACPI_FUNCTION_TRACE (ExDumpObjectDescriptor);


1055     if (!ObjDesc)
1056     {
1057         return_VOID;
1058     }

1060     if (!Flags)
1061     {
1062         /* Check if debug output enabled */

1064         if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_OBJECTS, _COMPONENT))
1080         if (!((ACPI_LV_OBJECTS & AcpiDbgLevel) && (_COMPONENT & AcpiDbgLayer)))
1065         {
1066             return_VOID;
1067         }
1068     }

1070     if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) == ACPI_DESC_TYPE_NAMED)
1071     {
1072         AcpiExDumpNamespaceNode ((ACPI_NAMESPACE_NODE *) ObjDesc, Flags);

1074         AcpiOsPrintf ("\nAttached Object (%p):\n",
1075             ((ACPI_NAMESPACE_NODE *) ObjDesc)->Object);

1077         AcpiExDumpObjectDescriptor (
1078             ((ACPI_NAMESPACE_NODE *) ObjDesc)->Object, Flags);
1079         return_VOID;
1080     }

1082     if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND)
1083     {
1084         AcpiOsPrintf (
1085             "ExDumpObjectDescriptor: %p is not an ACPI operand object: [%s]\n",
1086             ObjDesc, AcpiUtGetDescriptorName (ObjDesc));
1087         return_VOID;
1088     }

1090     if (ObjDesc->Common.Type > ACPI_TYPE_NS_NODE_MAX)
1091     {
1092         return_VOID;
1093     }

1095     /* Common Fields */

1097     AcpiExDumpObject (ObjDesc, AcpiExDumpCommon);

1099     /* Object-specific fields */

1101     AcpiExDumpObject (ObjDesc, AcpiExDumpInfo[ObjDesc->Common.Type]);
1102     return_VOID;
1103 }

1105 #endif
```

****
13428 Thu Dec 26 13:49:03 2013
new/usr/src/common/acpica/components/executer/exfield.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exfield - ACPI AML (p-code) execution - field manipulation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EXFIELD_C__

#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exfield")


/*******************************************************************************
 *
 * FUNCTION:    AcpiExReadDataFromField
```



```
 *
 * PARAMETERS:  WalkState           - Current execution state
 *              ObjDesc             - The named field
 *              RetBufferDesc       - Where the return data object is stored
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Read from a named field. Returns either an Integer or a
 *              Buffer, depending on the size of the field.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExReadDataFromField (
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     **RetBufferDesc)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *BufferDesc;
    ACPI_SIZE               Length;
    void                    *Buffer;
    UINT32                  Function;


    ACPI_FUNCTION_TRACE_PTR (ExReadDataFromField, ObjDesc);


    /* Parameter validation */

    if (!ObjDesc)
    {
        return_ACPI_STATUS (AE_AML_NO_OPERAND);
    }
    if (!RetBufferDesc)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (ObjDesc->Common.Type == ACPI_TYPE_BUFFER_FIELD)
    {
        /*
         * If the BufferField arguments have not been previously evaluated,
         * evaluate them now and save the results.
         */
        if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
        {
            Status = AcpiDsGetBufferFieldArguments (ObjDesc);
            if (ACPI_FAILURE (Status))
            {
                return_ACPI_STATUS (Status);
            }
        }
    }
    else if ((ObjDesc->Common.Type == ACPI_TYPE_LOCAL_REGION_FIELD) &&
             (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS |
              ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS |
              ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_IPMI))
    {
        /*
         * This is an SMBus, GSBus or IPMI read. We must create a buffer to hold
         * This is an SMBus or IPMI read. We must create a buffer to hold
         * the data and then directly access the region handler.
         *
         * Note: SMBus and GSBus protocol value is passed in upper 16-bits of Fu
         * Note: Smbus protocol value is passed in upper 16-bits of Function
```

```
124          */
125         if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS)
126         {
127             Length = ACPI_SMBUS_BUFFER_SIZE;
128             Function = ACPI_READ | (ObjDesc->Field.Attribute << 16);
129         }
130         else if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBU
131         {
132             Length = ACPI_GSBUS_BUFFER_SIZE;
133             Function = ACPI_READ | (ObjDesc->Field.Attribute << 16);
134         }
135         else /* IPMI */
136         {
137             Length = ACPI_IPMI_BUFFER_SIZE;
138             Function = ACPI_READ;
139         }
140
141         BufferDesc = AcpiUtCreateBufferObject (Length);
142         if (!BufferDesc)
143         {
144             return_ACPI_STATUS (AE_NO_MEMORY);
145         }
146
147         /* Lock entire transaction if requested */
148
149         AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);
150
151         /* Call the region handler for the read */
152
153         Status = AcpiExAccessRegion (ObjDesc, 0,
154                     ACPI_CAST_PTR (UINT64, BufferDesc->Buffer.Pointer),
155                     Function);
156         AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);
157         goto Exit;
158     }
159
160     /*
161      * Allocate a buffer for the contents of the field.
162      *
163      * If the field is larger than the current integer width, create
164      * a BUFFER to hold it. Otherwise, use an INTEGER. This allows
165      * the use of arithmetic operators on the returned value if the
166      * field size is equal or smaller than an Integer.
167      *
168      * Note: Field.length is in bits.
169      */
170     Length = (ACPI_SIZE) ACPI_ROUND_BITS_UP_TO_BYTES (ObjDesc->Field.BitLength);
171     if (Length > AcpiGbl_IntegerByteWidth)
172     {
173         /* Field is too large for an Integer, create a Buffer instead */
174
175         BufferDesc = AcpiUtCreateBufferObject (Length);
176         if (!BufferDesc)
177         {
178             return_ACPI_STATUS (AE_NO_MEMORY);
179         }
180         Buffer = BufferDesc->Buffer.Pointer;
181     }
182     else
183     {
184         /* Field will fit within an Integer (normal case) */
185
186         BufferDesc = AcpiUtCreateIntegerObject ((UINT64) 0);
187         if (!BufferDesc)
188         {
189             return_ACPI_STATUS (AE_NO_MEMORY);
```



```
190         }
191
192         Length = AcpiGbl_IntegerByteWidth;
193         Buffer = &BufferDesc->Integer.Value;
194     }
195
196     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
197         "FieldRead [TO]:   Obj %p, Type %X, Buf %p, ByteLen %X\n",
198         ObjDesc, ObjDesc->Common.Type, Buffer, (UINT32) Length));
199     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
200         "FieldRead [FROM]: BitLen %X, BitOff %X, ByteOff %X\n",
201         ObjDesc->CommonField.BitLength,
202         ObjDesc->CommonField.StartFieldBitOffset,
203         ObjDesc->CommonField.BaseByteOffset));
204
205     /* Lock entire transaction if requested */
206
207     AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);
208
209     /* Read from the field */
210
211     Status = AcpiExExtractFromField (ObjDesc, Buffer, (UINT32) Length);
212     AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);
213
214
215 Exit:
216     if (ACPI_FAILURE (Status))
217     {
218         AcpiUtRemoveReference (BufferDesc);
219     }
220     else
221     {
222         *RetBufferDesc = BufferDesc;
223     }
224
225     return_ACPI_STATUS (Status);
226 }
227
228
229 /*******************************************************************************
230  *
231  * FUNCTION:    AcpiExWriteDataToField
232  *
233  * PARAMETERS:  SourceDesc          - Contains data to write
234  *              ObjDesc             - The named field
235  *              ResultDesc          - Where the return value is returned, if any
236  *
237  * RETURN:      Status
238  *
239  * DESCRIPTION: Write to a named field
240  *
241  ******************************************************************************/
242
243 ACPI_STATUS
244 AcpiExWriteDataToField (
245     ACPI_OPERAND_OBJECT     *SourceDesc,
246     ACPI_OPERAND_OBJECT     *ObjDesc,
247     ACPI_OPERAND_OBJECT     **ResultDesc)
248 {
249     ACPI_STATUS             Status;
250     UINT32                  Length;
251     void                    *Buffer;
252     ACPI_OPERAND_OBJECT     *BufferDesc;
253     UINT32                  Function;
```

```
256     ACPI_FUNCTION_TRACE_PTR (ExWriteDataToField, ObjDesc);


259     /* Parameter validation */

261     if (!SourceDesc || !ObjDesc)
262     {
263         return_ACPI_STATUS (AE_AML_NO_OPERAND);
264     }

266     if (ObjDesc->Common.Type == ACPI_TYPE_BUFFER_FIELD)
267     {
268         /*
269          * If the BufferField arguments have not been previously evaluated,
270          * evaluate them now and save the results.
271          */
272         if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
273         {
274             Status = AcpiDsGetBufferFieldArguments (ObjDesc);
275             if (ACPI_FAILURE (Status))
276             {
277                 return_ACPI_STATUS (Status);
278             }
279         }
280     }
281     else if ((ObjDesc->Common.Type == ACPI_TYPE_LOCAL_REGION_FIELD) &&
282               (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS |
283                ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS |
284                ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_IPMI))
285     {
286         /*
287          * This is an SMBus, GSBus or IPMI write. We will bypass the entire fiel
280          * This is an SMBus or IPMI write. We will bypass the entire field
288          * mechanism and handoff the buffer directly to the handler. For
289          * these address spaces, the buffer is bi-directional; on a write,
290          * return data is returned in the same buffer.
291          *
292          * Source must be a buffer of sufficient size:
293          * ACPI_SMBUS_BUFFER_SIZE, ACPI_GSBUS_BUFFER_SIZE, or ACPI_IPMI_BUFFER_S
286          * ACPI_SMBUS_BUFFER_SIZE or ACPI_IPMI_BUFFER_SIZE.
294          *
295          * Note: SMBus and GSBus protocol type is passed in upper 16-bits of Fun
288          * Note: SMBus protocol type is passed in upper 16-bits of Function
296          */
297         if (SourceDesc->Common.Type != ACPI_TYPE_BUFFER)
298         {
299             ACPI_ERROR ((AE_INFO,
300                 "SMBus/IPMI/GenericSerialBus write requires Buffer, found type %
293                 "SMBus or IPMI write requires Buffer, found type %s",
301                 AcpiUtGetObjectTypeName (SourceDesc)));

303             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
304         }

306         if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS)
307         {
308             Length = ACPI_SMBUS_BUFFER_SIZE;
309             Function = ACPI_WRITE | (ObjDesc->Field.Attribute << 16);
310         }
311         else if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBU
312         {
313             Length = ACPI_GSBUS_BUFFER_SIZE;
314             Function = ACPI_WRITE | (ObjDesc->Field.Attribute << 16);
315         }
316         else /* IPMI */
317         {
```



```
318             Length = ACPI_IPMI_BUFFER_SIZE;
319             Function = ACPI_WRITE;
320         }

322         if (SourceDesc->Buffer.Length < Length)
323         {
324             ACPI_ERROR ((AE_INFO,
325                 "SMBus/IPMI/GenericSerialBus write requires Buffer of length %u,
313                 "SMBus or IPMI write requires Buffer of length %u, found length
326                 Length, SourceDesc->Buffer.Length));

328             return_ACPI_STATUS (AE_AML_BUFFER_LIMIT);
329         }

331         /* Create the bi-directional buffer */

333         BufferDesc = AcpiUtCreateBufferObject (Length);
334         if (!BufferDesc)
335         {
336             return_ACPI_STATUS (AE_NO_MEMORY);
337         }

339         Buffer = BufferDesc->Buffer.Pointer;
340         ACPI_MEMCPY (Buffer, SourceDesc->Buffer.Pointer, Length);

342         /* Lock entire transaction if requested */

344         AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);

346         /*
347          * Perform the write (returns status and perhaps data in the
348          * same buffer)
349          */
350         Status = AcpiExAccessRegion (ObjDesc, 0,
351                     (UINT64 *) Buffer, Function);
352         AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);

354         *ResultDesc = BufferDesc;
355         return_ACPI_STATUS (Status);
356     }

358     /* Get a pointer to the data to be written */

360     switch (SourceDesc->Common.Type)
361     {
362     case ACPI_TYPE_INTEGER:

364         Buffer = &SourceDesc->Integer.Value;
365         Length = sizeof (SourceDesc->Integer.Value);
366         break;

368     case ACPI_TYPE_BUFFER:

370         Buffer = SourceDesc->Buffer.Pointer;
371         Length = SourceDesc->Buffer.Length;
372         break;

374     case ACPI_TYPE_STRING:

376         Buffer = SourceDesc->String.Pointer;
377         Length = SourceDesc->String.Length;
378         break;

380     default:

382         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
```

```
383     }

385     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
386         "FieldWrite [FROM]: Obj %p (%s:%X), Buf %p, ByteLen %X\n",
387         SourceDesc, AcpiUtGetTypeName (SourceDesc->Common.Type),
388         SourceDesc->Common.Type, Buffer, Length));

390     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
391         "FieldWrite [TO]:   Obj %p (%s:%X), BitLen %X, BitOff %X, ByteOff %X\n",
392         ObjDesc, AcpiUtGetTypeName (ObjDesc->Common.Type),
393         ObjDesc->Common.Type,
394         ObjDesc->CommonField.BitLength,
395         ObjDesc->CommonField.StartFieldBitOffset,
396         ObjDesc->CommonField.BaseByteOffset));

398     /* Lock entire transaction if requested */

400     AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);

402     /* Write to the field */

404     Status = AcpiExInsertIntoField (ObjDesc, Buffer, Length);
405     AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);

407     return_ACPI_STATUS (Status);
408 }
```

```
**********************************************************
   33386 Thu Dec 26 13:49:03 2013
new/usr/src/common/acpica/components/executer/exfldio.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exfldio - Aml Field I/O
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EXFLDIO_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"
#include "acevents.h"
#include "acdispat.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exfldio")

/* Local prototypes */
```



```
static ACPI_STATUS
AcpiExFieldDatumIo (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  FieldDatumByteOffset,
    UINT64                  *Value,
    UINT32                  ReadWrite);

static BOOLEAN
AcpiExRegisterOverflow (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT64                  Value);

static ACPI_STATUS
AcpiExSetupRegion (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  FieldDatumByteOffset);


/*******************************************************************************
 *
 * FUNCTION:    AcpiExSetupRegion
 *
 * PARAMETERS:  ObjDesc                 - Field to be read or written
 *              FieldDatumByteOffset    - Byte offset of this datum within the
 *                                        parent field
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Common processing for AcpiExExtractFromField and
 *              AcpiExInsertIntoField. Initialize the Region if necessary and
 *              validate the request.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiExSetupRegion (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  FieldDatumByteOffset)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_OPERAND_OBJECT     *RgnDesc;
    UINT8                   SpaceId;


    ACPI_FUNCTION_TRACE_U32 (ExSetupRegion, FieldDatumByteOffset);


    RgnDesc = ObjDesc->CommonField.RegionObj;

    /* We must have a valid region */

    if (RgnDesc->Common.Type != ACPI_TYPE_REGION)
    {
        ACPI_ERROR ((AE_INFO, "Needed Region, found type 0x%X (%s)",
            RgnDesc->Common.Type,
            AcpiUtGetObjectTypeName (RgnDesc)));

        return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
    }

    SpaceId = RgnDesc->Region.SpaceId;

    /* Validate the Space ID */

    if (!AcpiIsValidSpaceId (SpaceId))
    {
```

```
126         ACPI_ERROR ((AE_INFO, "Invalid/unknown Address Space ID: 0x%2.2X", Space
127         return_ACPI_STATUS (AE_AML_INVALID_SPACE_ID);
128     }

130     /*
131      * If the Region Address and Length have not been previously evaluated,
132      * evaluate them now and save the results.
133      */
134     if (!(RgnDesc->Common.Flags & AOPOBJ_DATA_VALID))
135     {
136         Status = AcpiDsGetRegionArguments (RgnDesc);
137         if (ACPI_FAILURE (Status))
138         {
139             return_ACPI_STATUS (Status);
140         }
141     }

143     /*
144      * Exit now for SMBus, GSBus or IPMI address space, it has a non-linear
133      * Exit now for SMBus or IPMI address space, it has a non-linear
145      * address space and the request cannot be directly validated
146      */
147     if (SpaceId == ACPI_ADR_SPACE_SMBUS ||
148         SpaceId == ACPI_ADR_SPACE_GSBUS ||
149         SpaceId == ACPI_ADR_SPACE_IPMI)
136     if (RgnDesc->Region.SpaceId == ACPI_ADR_SPACE_SMBUS ||
137         RgnDesc->Region.SpaceId == ACPI_ADR_SPACE_IPMI)
150     {
151         /* SMBus or IPMI has a non-linear address space */

153         return_ACPI_STATUS (AE_OK);
154     }

156 #ifdef ACPI_UNDER_DEVELOPMENT
157     /*
158      * If the Field access is AnyAcc, we can now compute the optimal
159      * access (because we know know the length of the parent region)
160      */
161     if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
162     {
163         if (ACPI_FAILURE (Status))
164         {
165             return_ACPI_STATUS (Status);
166         }
167     }
168 #endif

170     /*
171      * Validate the request. The entire request from the byte offset for a
172      * length of one field datum (access width) must fit within the region.
173      * (Region length is specified in bytes)
174      */
175     if (RgnDesc->Region.Length <
176             (ObjDesc->CommonField.BaseByteOffset + FieldDatumByteOffset +
177             ObjDesc->CommonField.AccessByteWidth))
178     {
179         if (AcpiGbl_EnableInterpreterSlack)
180         {
181             /*
182              * Slack mode only:  We will go ahead and allow access to this
183              * field if it is within the region length rounded up to the next
184              * access width boundary. ACPI_SIZE cast for 64-bit compile.
185              */
186             if (ACPI_ROUND_UP (RgnDesc->Region.Length,
187                     ObjDesc->CommonField.AccessByteWidth) >=
188                 ((ACPI_SIZE) ObjDesc->CommonField.BaseByteOffset +
```



```
189                     ObjDesc->CommonField.AccessByteWidth +
190                     FieldDatumByteOffset))
191             {
192                 return_ACPI_STATUS (AE_OK);
193             }
194         }

196         if (RgnDesc->Region.Length < ObjDesc->CommonField.AccessByteWidth)
197         {
198             /*
199              * This is the case where the AccessType (AccWord, etc.) is wider
200              * than the region itself. For example, a region of length one
201              * byte, and a field with Dword access specified.
202              */
203             ACPI_ERROR ((AE_INFO,
204                 "Field [%4.4s] access width (%u bytes) too large for region [%4.
205                 AcpiUtGetNodeName (ObjDesc->CommonField.Node),
206                 ObjDesc->CommonField.AccessByteWidth,
207                 AcpiUtGetNodeName (RgnDesc->Region.Node),
208                 RgnDesc->Region.Length));
209         }

211         /*
212          * Offset rounded up to next multiple of field width
213          * exceeds region length, indicate an error
214          */
215         ACPI_ERROR ((AE_INFO,
216             "Field [%4.4s] Base+Offset+Width %u+%u+%u is beyond end of region [%
217             AcpiUtGetNodeName (ObjDesc->CommonField.Node),
218             ObjDesc->CommonField.BaseByteOffset,
219             FieldDatumByteOffset, ObjDesc->CommonField.AccessByteWidth,
220             AcpiUtGetNodeName (RgnDesc->Region.Node),
221             RgnDesc->Region.Length));

223         return_ACPI_STATUS (AE_AML_REGION_LIMIT);
224     }

226     return_ACPI_STATUS (AE_OK);
227 }


230 /*******************************************************************************
231  *
232  * FUNCTION:    AcpiExAccessRegion
233  *
234  * PARAMETERS:  ObjDesc                 - Field to be read
235  *              FieldDatumByteOffset    - Byte offset of this datum within the
236  *                                        parent field
237  *              Value                   - Where to store value (must at least
238  *                                        64 bits)
239  *              Function                - Read or Write flag plus other region-
240  *                                        dependent flags
241  *
242  * RETURN:      Status
243  *
244  * DESCRIPTION: Read or Write a single field datum to an Operation Region.
245  *
246  ******************************************************************************/

248 ACPI_STATUS
249 AcpiExAccessRegion (
250     ACPI_OPERAND_OBJECT     *ObjDesc,
251     UINT32                  FieldDatumByteOffset,
252     UINT64                  *Value,
253     UINT32                  Function)
254 {
```

```
255     ACPI_STATUS             Status;
256     ACPI_OPERAND_OBJECT     *RgnDesc;
257     UINT32                  RegionOffset;


260     ACPI_FUNCTION_TRACE (ExAccessRegion);


263     /*
264      * Ensure that the region operands are fully evaluated and verify
265      * the validity of the request
266      */
267     Status = AcpiExSetupRegion (ObjDesc, FieldDatumByteOffset);
268     if (ACPI_FAILURE (Status))
269     {
270         return_ACPI_STATUS (Status);
271     }

273     /*
274      * The physical address of this field datum is:
275      *
276      * 1) The base of the region, plus
277      * 2) The base offset of the field, plus
278      * 3) The current offset into the field
279      */
280     RgnDesc = ObjDesc->CommonField.RegionObj;
281     RegionOffset =
282         ObjDesc->CommonField.BaseByteOffset +
283         FieldDatumByteOffset;

285     if ((Function & ACPI_IO_MASK) == ACPI_READ)
286     {
287         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD, "[READ]"));
288     }
289     else
290     {
291         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD, "[WRITE]"));
292     }

294     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_BFIELD,
295         " Region [%s:%X], Width %X, ByteBase %X, Offset %X at %p\n",
296         AcpiUtGetRegionName (RgnDesc->Region.SpaceId),
297         RgnDesc->Region.SpaceId,
298         ObjDesc->CommonField.AccessByteWidth,
299         ObjDesc->CommonField.BaseByteOffset,
300         FieldDatumByteOffset,
301         ACPI_CAST_PTR (void, (RgnDesc->Region.Address + RegionOffset))));

303     /* Invoke the appropriate AddressSpace/OpRegion handler */

305     Status = AcpiEvAddressSpaceDispatch (RgnDesc, ObjDesc,
306                 Function, RegionOffset,
293     Status = AcpiEvAddressSpaceDispatch (RgnDesc, Function, RegionOffset,
307                 ACPI_MUL_8 (ObjDesc->CommonField.AccessByteWidth), Value);

309     if (ACPI_FAILURE (Status))
310     {
311         if (Status == AE_NOT_IMPLEMENTED)
312         {
313             ACPI_ERROR ((AE_INFO,
314                 "Region %s (ID=%u) not implemented",
315                 AcpiUtGetRegionName (RgnDesc->Region.SpaceId),
316                 RgnDesc->Region.SpaceId));
317         }
318         else if (Status == AE_NOT_EXIST)
319         {
```



```
320             ACPI_ERROR ((AE_INFO,
321                 "Region %s (ID=%u) has no handler",
322                 AcpiUtGetRegionName (RgnDesc->Region.SpaceId),
323                 RgnDesc->Region.SpaceId));
324         }
325     }

327     return_ACPI_STATUS (Status);
328 }


331 /*******************************************************************************
332  *
333  * FUNCTION:    AcpiExRegisterOverflow
334  *
335  * PARAMETERS:  ObjDesc                 - Register(Field) to be written
336  *              Value                   - Value to be stored
337  *
338  * RETURN:      TRUE if value overflows the field, FALSE otherwise
339  *
340  * DESCRIPTION: Check if a value is out of range of the field being written.
341  *              Used to check if the values written to Index and Bank registers
342  *              are out of range. Normally, the value is simply truncated
343  *              to fit the field, but this case is most likely a serious
344  *              coding error in the ASL.
345  *
346  ******************************************************************************/

348 static BOOLEAN
349 AcpiExRegisterOverflow (
350     ACPI_OPERAND_OBJECT     *ObjDesc,
351     UINT64                  Value)
352 {

354     if (ObjDesc->CommonField.BitLength >= ACPI_INTEGER_BIT_SIZE)
355     {
356         /*
357          * The field is large enough to hold the maximum integer, so we can
358          * never overflow it.
359          */
360         return (FALSE);
361     }

363     if (Value >= ((UINT64) 1 << ObjDesc->CommonField.BitLength))
364     {
365         /*
366          * The Value is larger than the maximum value that can fit into
367          * the register.
368          */
369         ACPI_ERROR ((AE_INFO,
370             "Index value 0x%8.8X%8.8X overflows field width 0x%X",
371             ACPI_FORMAT_UINT64 (Value),
372             ObjDesc->CommonField.BitLength));

374         return (TRUE);
375     }

377     /* The Value will fit into the field with no truncation */

379     return (FALSE);
380 }


383 /*******************************************************************************
384  *
385  * FUNCTION:    AcpiExFieldDatumIo
```

```
 386  *
 387  * PARAMETERS:  ObjDesc                 - Field to be read
 388  *              FieldDatumByteOffset    - Byte offset of this datum within the
 389  *                                        parent field
 390  *              Value                   - Where to store value (must be 64 bits)
 391  *              ReadWrite               - Read or Write flag
 392  *
 393  * RETURN:      Status
 394  *
 395  * DESCRIPTION: Read or Write a single datum of a field. The FieldType is
 396  *              demultiplexed here to handle the different types of fields
 397  *              (BufferField, RegionField, IndexField, BankField)
 398  *
 399  *****************************************************************************/

 401 static ACPI_STATUS
 402 AcpiExFieldDatumIo (
 403     ACPI_OPERAND_OBJECT     *ObjDesc,
 404     UINT32                  FieldDatumByteOffset,
 405     UINT64                  *Value,
 406     UINT32                  ReadWrite)
 407 {
 408     ACPI_STATUS             Status;
 409     UINT64                  LocalValue;


 412     ACPI_FUNCTION_TRACE_U32 (ExFieldDatumIo, FieldDatumByteOffset);


 415     if (ReadWrite == ACPI_READ)
 416     {
 417         if (!Value)
 418         {
 419             LocalValue = 0;

 421             /* To support reads without saving return value */
 422             Value = &LocalValue;
 423         }

 425         /* Clear the entire return buffer first, [Very Important!] */

 427         *Value = 0;
 428     }

 430     /*
 431      * The four types of fields are:
 432      *
 433      * BufferField - Read/write from/to a Buffer
 434      * RegionField - Read/write from/to a Operation Region.
 435      * BankField   - Write to a Bank Register, then read/write from/to an
 436      *               OperationRegion
 437      * IndexField  - Write to an Index Register, then read/write from/to a
 438      *               Data Register
 439      */
 440     switch (ObjDesc->Common.Type)
 441     {
 442     case ACPI_TYPE_BUFFER_FIELD:
 443         /*
 444          * If the BufferField arguments have not been previously evaluated,
 445          * evaluate them now and save the results.
 446          */
 447         if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
 448         {
 449             Status = AcpiDsGetBufferFieldArguments (ObjDesc);
 450             if (ACPI_FAILURE (Status))
 451             {
```



```
 452                 return_ACPI_STATUS (Status);
 453             }
 454         }

 456         if (ReadWrite == ACPI_READ)
 457         {
 458             /*
 459              * Copy the data from the source buffer.
 460              * Length is the field width in bytes.
 461              */
 462             ACPI_MEMCPY (Value,
 463                 (ObjDesc->BufferField.BufferObj)->Buffer.Pointer +
 464                     ObjDesc->BufferField.BaseByteOffset +
 465                     FieldDatumByteOffset,
 466                 ObjDesc->CommonField.AccessByteWidth);
 467         }
 468         else
 469         {
 470             /*
 471              * Copy the data to the target buffer.
 472              * Length is the field width in bytes.
 473              */
 474             ACPI_MEMCPY ((ObjDesc->BufferField.BufferObj)->Buffer.Pointer +
 475                 ObjDesc->BufferField.BaseByteOffset +
 476                 FieldDatumByteOffset,
 477                 Value, ObjDesc->CommonField.AccessByteWidth);
 478         }

 480         Status = AE_OK;
 481         break;


 483     case ACPI_TYPE_LOCAL_BANK_FIELD:

 484         /*
 485          * Ensure that the BankValue is not beyond the capacity of
 486          * the register
 487          */
 488         if (AcpiExRegisterOverflow (ObjDesc->BankField.BankObj,
 489                 (UINT64) ObjDesc->BankField.Value))
 490         {
 491             return_ACPI_STATUS (AE_AML_REGISTER_LIMIT);
 492         }

 494         /*
 495          * For BankFields, we must write the BankValue to the BankRegister
 496          * (itself a RegionField) before we can access the data.
 497          */
 498         Status = AcpiExInsertIntoField (ObjDesc->BankField.BankObj,
 499                     &ObjDesc->BankField.Value,
 500                     sizeof (ObjDesc->BankField.Value));
 501         if (ACPI_FAILURE (Status))
 502         {
 503             return_ACPI_STATUS (Status);
 504         }

 506         /*
 507          * Now that the Bank has been selected, fall through to the
 508          * RegionField case and write the datum to the Operation Region
 509          */

 511         /*lint -fallthrough */


 513     case ACPI_TYPE_LOCAL_REGION_FIELD:
 514         /*
```

```
515          * For simple RegionFields, we just directly access the owning
516          * Operation Region.
517          */
518         Status = AcpiExAccessRegion (ObjDesc, FieldDatumByteOffset, Value,
519                     ReadWrite);
520         break;


522     case ACPI_TYPE_LOCAL_INDEX_FIELD:


523         /*
524          * Ensure that the IndexValue is not beyond the capacity of
525          * the register
526          */
527         if (AcpiExRegisterOverflow (ObjDesc->IndexField.IndexObj,
528                 (UINT64) ObjDesc->IndexField.Value))
529         {
530             return_ACPI_STATUS (AE_AML_REGISTER_LIMIT);
531         }

533         /* Write the index value to the IndexRegister (itself a RegionField) */

535         FieldDatumByteOffset += ObjDesc->IndexField.Value;

537         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
538             "Write to Index Register: Value %8.8X\n",
539             FieldDatumByteOffset));

541         Status = AcpiExInsertIntoField (ObjDesc->IndexField.IndexObj,
542                     &FieldDatumByteOffset,
543                     sizeof (FieldDatumByteOffset));
544         if (ACPI_FAILURE (Status))
545         {
546             return_ACPI_STATUS (Status);
547         }

549         if (ReadWrite == ACPI_READ)
550         {
551             /* Read the datum from the DataRegister */

553             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
554                 "Read from Data Register\n"));

556             Status = AcpiExExtractFromField (ObjDesc->IndexField.DataObj,
557                         Value, sizeof (UINT64));
558         }
559         else
560         {
561             /* Write the datum to the DataRegister */

563             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
564                 "Write to Data Register: Value %8.8X%8.8X\n",
565                 ACPI_FORMAT_UINT64 (*Value)));

567             Status = AcpiExInsertIntoField (ObjDesc->IndexField.DataObj,
568                         Value, sizeof (UINT64));
569         }
570         break;


572     default:

574         ACPI_ERROR ((AE_INFO, "Wrong object type in field I/O %u",
575             ObjDesc->Common.Type));
576         Status = AE_AML_INTERNAL;
```



```
577         break;
578     }

580     if (ACPI_SUCCESS (Status))
581     {
582         if (ReadWrite == ACPI_READ)
583         {
584             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
585                 "Value Read %8.8X%8.8X, Width %u\n",
586                 ACPI_FORMAT_UINT64 (*Value),
587                 ObjDesc->CommonField.AccessByteWidth));
588         }
589         else
590         {
591             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
592                 "Value Written %8.8X%8.8X, Width %u\n",
593                 ACPI_FORMAT_UINT64 (*Value),
594                 ObjDesc->CommonField.AccessByteWidth));
595         }
596     }

598     return_ACPI_STATUS (Status);
599 }
```

unchanged_portion_omitted_

```
707 /*******************************************************************************
708  *
709  * FUNCTION:    AcpiExExtractFromField
710  *
711  * PARAMETERS:  ObjDesc             - Field to be read
712  *              Buffer              - Where to store the field data
713  *              BufferLength        - Length of Buffer
714  *
715  * RETURN:      Status
716  *
717  * DESCRIPTION: Retrieve the current value of the given field
718  *
719  ******************************************************************************/

721 ACPI_STATUS
722 AcpiExExtractFromField (
723     ACPI_OPERAND_OBJECT     *ObjDesc,
724     void                    *Buffer,
725     UINT32                  BufferLength)
726 {
727     ACPI_STATUS             Status;
728     UINT64                  RawDatum;
729     UINT64                  MergedDatum;
730     UINT32                  FieldOffset = 0;
731     UINT32                  BufferOffset = 0;
732     UINT32                  BufferTailBits;
733     UINT32                  DatumCount;
734     UINT32                  FieldDatumCount;
735     UINT32                  AccessBitWidth;
736     UINT32                  i;


739     ACPI_FUNCTION_TRACE (ExExtractFromField);


742     /* Validate target buffer and clear it */

744     if (BufferLength <
745         ACPI_ROUND_BITS_UP_TO_BYTES (ObjDesc->CommonField.BitLength))
746     {
```

```
747         ACPI_ERROR ((AE_INFO,
748             "Field size %u (bits) is too large for buffer (%u)",
749             ObjDesc->CommonField.BitLength, BufferLength));
750
751         return_ACPI_STATUS (AE_BUFFER_OVERFLOW);
752     }
753
754     ACPI_MEMSET (Buffer, 0, BufferLength);
755     AccessBitWidth = ACPI_MUL_8 (ObjDesc->CommonField.AccessByteWidth);
756
757     /* Handle the simple case here */
758
759     if ((ObjDesc->CommonField.StartFieldBitOffset == 0) &&
760         (ObjDesc->CommonField.BitLength == AccessBitWidth))
761     {
762         if (BufferLength >= sizeof (UINT64))
763         {
764             Status = AcpiExFieldDatumIo (ObjDesc, 0, Buffer, ACPI_READ);
765         }
766         else
767         {
768             /* Use RawDatum (UINT64) to handle buffers < 64 bits */
769
770             Status = AcpiExFieldDatumIo (ObjDesc, 0, &RawDatum, ACPI_READ);
771             ACPI_MEMCPY (Buffer, &RawDatum, BufferLength);
772         }
773
774         return_ACPI_STATUS (Status);
775     }
776
777 /* TBD: Move to common setup code */
778
779     /* Field algorithm is limited to sizeof(UINT64), truncate if needed */
780
781     if (ObjDesc->CommonField.AccessByteWidth > sizeof (UINT64))
782     {
783         ObjDesc->CommonField.AccessByteWidth = sizeof (UINT64);
784         AccessBitWidth = sizeof (UINT64) * 8;
785     }
786
787     /* Compute the number of datums (access width data items) */
788
789     DatumCount = ACPI_ROUND_UP_TO (
790         ObjDesc->CommonField.BitLength, AccessBitWidth);
791
792     FieldDatumCount = ACPI_ROUND_UP_TO (
793         ObjDesc->CommonField.BitLength +
794         ObjDesc->CommonField.StartFieldBitOffset, AccessBitWidth);
795
796     /* Priming read from the field */
797
798     Status = AcpiExFieldDatumIo (ObjDesc, FieldOffset, &RawDatum, ACPI_READ);
799     if (ACPI_FAILURE (Status))
800     {
801         return_ACPI_STATUS (Status);
802     }
803     MergedDatum = RawDatum >> ObjDesc->CommonField.StartFieldBitOffset;
804
805     /* Read the rest of the field */
806
807     for (i = 1; i < FieldDatumCount; i++)
808     {
809         /* Get next input datum from the field */
810
811         FieldOffset += ObjDesc->CommonField.AccessByteWidth;
812         Status = AcpiExFieldDatumIo (ObjDesc, FieldOffset,
```



```
813                                      &RawDatum, ACPI_READ);
814         if (ACPI_FAILURE (Status))
815         {
816             return_ACPI_STATUS (Status);
817         }
818
819         /*
820          * Merge with previous datum if necessary.
821          *
822          * Note: Before the shift, check if the shift value will be larger than
823          * the integer size. If so, there is no need to perform the operation.
824          * This avoids the differences in behavior between different compilers
825          * concerning shift values larger than the target data width.
826          */
827         if (AccessBitWidth - ObjDesc->CommonField.StartFieldBitOffset <
828             ACPI_INTEGER_BIT_SIZE)
829         {
830             MergedDatum |= RawDatum <<
831                 (AccessBitWidth - ObjDesc->CommonField.StartFieldBitOffset);
832         }
833
834         if (i == DatumCount)
835         {
836             break;
837         }
838
839         /* Write merged datum to target buffer */
840
841         ACPI_MEMCPY (((char *) Buffer) + BufferOffset, &MergedDatum,
842             ACPI_MIN(ObjDesc->CommonField.AccessByteWidth,
843                 BufferLength - BufferOffset));
844
845         BufferOffset += ObjDesc->CommonField.AccessByteWidth;
846         MergedDatum = RawDatum >> ObjDesc->CommonField.StartFieldBitOffset;
847     }
848
849     /* Mask off any extra bits in the last datum */
850
851     BufferTailBits = ObjDesc->CommonField.BitLength % AccessBitWidth;
852     if (BufferTailBits)
853     {
854         MergedDatum &= ACPI_MASK_BITS_ABOVE (BufferTailBits);
855     }
856
857     /* Write the last datum to the buffer */
858
859     ACPI_MEMCPY (((char *) Buffer) + BufferOffset, &MergedDatum,
860         ACPI_MIN(ObjDesc->CommonField.AccessByteWidth,
861             BufferLength - BufferOffset));
862
863     return_ACPI_STATUS (AE_OK);
864 }
______unchanged_portion_omitted_
```

****
23075 Thu Dec 26 13:49:04 2013
new/usr/src/common/acpica/components/executer/exmisc.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exmisc - ACPI AML (p-code) execution - specific opcodes
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXMISC_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"
#include "amlresrc.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exmisc")


/*******************************************************************************
 *
```



```
 * FUNCTION:    AcpiExGetObjectReference
 *
 * PARAMETERS:  ObjDesc             - Create a reference to this object
 *              ReturnDesc          - Where to store the reference
 *              WalkState           - Current state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Obtain and return a "reference" to the target object
 *              Common code for the RefOfOp and the CondRefOfOp.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExGetObjectReference (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     **ReturnDesc,
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_OPERAND_OBJECT     *ReferenceObj;
    ACPI_OPERAND_OBJECT     *ReferencedObj;


    ACPI_FUNCTION_TRACE_PTR (ExGetObjectReference, ObjDesc);


    *ReturnDesc = NULL;

    switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
    {
    case ACPI_DESC_TYPE_OPERAND:

        if (ObjDesc->Common.Type != ACPI_TYPE_LOCAL_REFERENCE)
        {
            return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
        }

        /*
         * Must be a reference to a Local or Arg
         */
        switch (ObjDesc->Reference.Class)
        {
        case ACPI_REFCLASS_LOCAL:
        case ACPI_REFCLASS_ARG:
        case ACPI_REFCLASS_DEBUG:

            /* The referenced object is the pseudo-node for the local/arg */

            ReferencedObj = ObjDesc->Reference.Object;
            break;

        default:

            ACPI_ERROR ((AE_INFO, "Unknown Reference Class 0x%2.2X",
                ObjDesc->Reference.Class));
            return_ACPI_STATUS (AE_AML_INTERNAL);
        }
        break;


    case ACPI_DESC_TYPE_NAMED:

        /*
         * A named reference that has already been resolved to a Node
         */
        ReferencedObj = ObjDesc;
```

```
123         break;


125     default:

127         ACPI_ERROR ((AE_INFO, "Invalid descriptor type 0x%X",
128             ACPI_GET_DESCRIPTOR_TYPE (ObjDesc)));
129         return_ACPI_STATUS (AE_TYPE);
130     }


133     /* Create a new reference object */

135     ReferenceObj = AcpiUtCreateInternalObject (ACPI_TYPE_LOCAL_REFERENCE);
136     if (!ReferenceObj)
137     {
138         return_ACPI_STATUS (AE_NO_MEMORY);
139     }

141     ReferenceObj->Reference.Class = ACPI_REFCLASS_REFOF;
142     ReferenceObj->Reference.Object = ReferencedObj;
143     *ReturnDesc = ReferenceObj;

145     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
146         "Object %p Type [%s], returning Reference %p\n",
147         ObjDesc, AcpiUtGetObjectTypeName (ObjDesc), *ReturnDesc));

149     return_ACPI_STATUS (AE_OK);
150 }
```

unchanged_portion_omitted_

```
245 /*******************************************************************************
246  *
247  * FUNCTION:    AcpiExDoConcatenate
248  *
249  * PARAMETERS:  Operand0            - First source object
250  *              Operand1            - Second source object
251  *              ActualReturnDesc    - Where to place the return object
252  *              WalkState           - Current walk state
253  *
254  * RETURN:      Status
255  *
256  * DESCRIPTION: Concatenate two objects OF THE SAME TYPE.
257  *
258  ******************************************************************************/

260 ACPI_STATUS
261 AcpiExDoConcatenate (
262     ACPI_OPERAND_OBJECT     *Operand0,
263     ACPI_OPERAND_OBJECT     *Operand1,
264     ACPI_OPERAND_OBJECT     **ActualReturnDesc,
265     ACPI_WALK_STATE         *WalkState)
266 {
267     ACPI_OPERAND_OBJECT     *LocalOperand1 = Operand1;
268     ACPI_OPERAND_OBJECT     *ReturnDesc;
269     char                    *NewBuf;
270     ACPI_STATUS             Status;


273     ACPI_FUNCTION_TRACE (ExDoConcatenate);


276     /*
277      * Convert the second operand if necessary. The first operand
278      * determines the type of the second operand, (See the Data Types
```



```
279      * section of the ACPI specification.)  Both object types are
280      * guaranteed to be either Integer/String/Buffer by the operand
281      * resolution mechanism.
282      */
283     switch (Operand0->Common.Type)
284     {
285     case ACPI_TYPE_INTEGER:

287         Status = AcpiExConvertToInteger (Operand1, &LocalOperand1, 16);
288         break;

290     case ACPI_TYPE_STRING:

292         Status = AcpiExConvertToString (Operand1, &LocalOperand1,
293                     ACPI_IMPLICIT_CONVERT_HEX);
294         break;

296     case ACPI_TYPE_BUFFER:

298         Status = AcpiExConvertToBuffer (Operand1, &LocalOperand1);
299         break;

301     default:

303         ACPI_ERROR ((AE_INFO, "Invalid object type: 0x%X",
304             Operand0->Common.Type));
305         Status = AE_AML_INTERNAL;
306     }

308     if (ACPI_FAILURE (Status))
309     {
310         goto Cleanup;
311     }

313     /*
314      * Both operands are now known to be the same object type
315      * (Both are Integer, String, or Buffer), and we can now perform the
316      * concatenation.
317      */

319     /*
320      * There are three cases to handle:
321      *
322      * 1) Two Integers concatenated to produce a new Buffer
323      * 2) Two Strings concatenated to produce a new String
324      * 3) Two Buffers concatenated to produce a new Buffer
325      */
326     switch (Operand0->Common.Type)
327     {
328     case ACPI_TYPE_INTEGER:

330         /* Result of two Integers is a Buffer */
331         /* Need enough buffer space for two integers */

333         ReturnDesc = AcpiUtCreateBufferObject ((ACPI_SIZE)
334                             ACPI_MUL_2 (AcpiGbl_IntegerByteWidth));
335         if (!ReturnDesc)
336         {
337             Status = AE_NO_MEMORY;
338             goto Cleanup;
339         }

341         NewBuf = (char *) ReturnDesc->Buffer.Pointer;

343         /* Copy the first integer, LSB first */
```

```
345         ACPI_MEMCPY (NewBuf, &Operand0->Integer.Value,
346                         AcpiGbl_IntegerByteWidth);
347
348         /* Copy the second integer (LSB first) after the first */
349
350         ACPI_MEMCPY (NewBuf + AcpiGbl_IntegerByteWidth,
351                         &LocalOperand1->Integer.Value,
352                         AcpiGbl_IntegerByteWidth);
353         break;
354
355     case ACPI_TYPE_STRING:
356
357         /* Result of two Strings is a String */
358
359         ReturnDesc = AcpiUtCreateStringObject (
360                         ((ACPI_SIZE) Operand0->String.Length +
361                         LocalOperand1->String.Length));
362         if (!ReturnDesc)
363         {
364             Status = AE_NO_MEMORY;
365             goto Cleanup;
366         }
367
368         NewBuf = ReturnDesc->String.Pointer;
369
370         /* Concatenate the strings */
371
372         ACPI_STRCPY (NewBuf, Operand0->String.Pointer);
373         ACPI_STRCPY (NewBuf + Operand0->String.Length,
374                         LocalOperand1->String.Pointer);
375         break;
376
377     case ACPI_TYPE_BUFFER:
378
379         /* Result of two Buffers is a Buffer */
380
381         ReturnDesc = AcpiUtCreateBufferObject (
382                         ((ACPI_SIZE) Operand0->Buffer.Length +
383                         LocalOperand1->Buffer.Length));
384         if (!ReturnDesc)
385         {
386             Status = AE_NO_MEMORY;
387             goto Cleanup;
388         }
389
390         NewBuf = (char *) ReturnDesc->Buffer.Pointer;
391
392         /* Concatenate the buffers */
393
394         ACPI_MEMCPY (NewBuf, Operand0->Buffer.Pointer,
395                         Operand0->Buffer.Length);
396         ACPI_MEMCPY (NewBuf + Operand0->Buffer.Length,
397                         LocalOperand1->Buffer.Pointer,
398                         LocalOperand1->Buffer.Length);
399         break;
400
401     default:
402
403         /* Invalid object type, should not happen here */
404
405         ACPI_ERROR ((AE_INFO, "Invalid object type: 0x%X",
406             Operand0->Common.Type));
407         Status =AE_AML_INTERNAL;
408         goto Cleanup;
409     }
```



```
410
411     *ActualReturnDesc = ReturnDesc;
412
413 Cleanup:
414     if (LocalOperand1 != Operand1)
415     {
416         AcpiUtRemoveReference (LocalOperand1);
417     }
418     return_ACPI_STATUS (Status);
419 }
420
421
422 /*******************************************************************************
423  *
424  * FUNCTION:    AcpiExDoMathOp
425  *
426  * PARAMETERS:  Opcode              - AML opcode
427  *              Integer0            - Integer operand #0
428  *              Integer1            - Integer operand #1
429  *
430  * RETURN:      Integer result of the operation
431  *
432  * DESCRIPTION: Execute a math AML opcode. The purpose of having all of the
433  *              math functions here is to prevent a lot of pointer dereferencing
434  *              to obtain the operands.
435  *
436  ******************************************************************************/
437
438 UINT64
439 AcpiExDoMathOp (
440     UINT16                  Opcode,
441     UINT64                  Integer0,
442     UINT64                  Integer1)
443 {
444
445     ACPI_FUNCTION_ENTRY ();
446
447
448     switch (Opcode)
449     {
450     case AML_ADD_OP:                /* Add (Integer0, Integer1, Result) */
451
452         return (Integer0 + Integer1);
453
454     case AML_BIT_AND_OP:            /* And (Integer0, Integer1, Result) */
455
456         return (Integer0 & Integer1);
457
458     case AML_BIT_NAND_OP:           /* NAnd (Integer0, Integer1, Result) */
459
460         return (~(Integer0 & Integer1));
461
462     case AML_BIT_OR_OP:             /* Or (Integer0, Integer1, Result) */
463
464         return (Integer0 | Integer1);
465
466     case AML_BIT_NOR_OP:            /* NOr (Integer0, Integer1, Result) */
467
468         return (~(Integer0 | Integer1));
469
470     case AML_BIT_XOR_OP:            /* XOr (Integer0, Integer1, Result) */
```

```
472         return (Integer0 ^ Integer1);


474     case AML_MULTIPLY_OP:           /* Multiply (Integer0, Integer1, Result) */

476         return (Integer0 * Integer1);


478     case AML_SHIFT_LEFT_OP:         /* ShiftLeft (Operand, ShiftCount, Result)*/

480         /*
481          * We need to check if the shiftcount is larger than the integer bit
482          * width since the behavior of this is not well-defined in the C languag
483          */
484         if (Integer1 >= AcpiGbl_IntegerBitWidth)
485         {
486             return (0);
487         }
488         return (Integer0 << Integer1);


490     case AML_SHIFT_RIGHT_OP:        /* ShiftRight (Operand, ShiftCount, Result)

492         /*
493          * We need to check if the shiftcount is larger than the integer bit
494          * width since the behavior of this is not well-defined in the C languag
495          */
496         if (Integer1 >= AcpiGbl_IntegerBitWidth)
497         {
498             return (0);
499         }
500         return (Integer0 >> Integer1);


502     case AML_SUBTRACT_OP:           /* Subtract (Integer0, Integer1, Result) */

504         return (Integer0 - Integer1);

506     default:

508         return (0);
509     }
510 }


513 /*******************************************************************************
514  *
515  * FUNCTION:    AcpiExDoLogicalNumericOp
516  *
517  * PARAMETERS:  Opcode              - AML opcode
518  *              Integer0            - Integer operand #0
519  *              Integer1            - Integer operand #1
520  *              LogicalResult       - TRUE/FALSE result of the operation
521  *
522  * RETURN:      Status
523  *
524  * DESCRIPTION: Execute a logical "Numeric" AML opcode. For these Numeric
525  *              operators (LAnd and LOr), both operands must be integers.
526  *
527  *              Note: cleanest machine code seems to be produced by the code
528  *              below, rather than using statements of the form:
529  *                  Result = (Integer0 && Integer1);
530  *
531  ******************************************************************************/

533 ACPI_STATUS
```



```
534 AcpiExDoLogicalNumericOp (
535     UINT16                  Opcode,
536     UINT64                  Integer0,
537     UINT64                  Integer1,
538     BOOLEAN                 *LogicalResult)
539 {
540     ACPI_STATUS             Status = AE_OK;
541     BOOLEAN                 LocalResult = FALSE;


544     ACPI_FUNCTION_TRACE (ExDoLogicalNumericOp);


547     switch (Opcode)
548     {
549     case AML_LAND_OP:               /* LAnd (Integer0, Integer1) */

551         if (Integer0 && Integer1)
552         {
553             LocalResult = TRUE;
554         }
555         break;

557     case AML_LOR_OP:                /* LOr (Integer0, Integer1) */

559         if (Integer0 || Integer1)
560         {
561             LocalResult = TRUE;
562         }
563         break;

565     default:

567         Status = AE_AML_INTERNAL;
568         break;
569     }

571     /* Return the logical result and status */

573     *LogicalResult = LocalResult;
574     return_ACPI_STATUS (Status);
575 }


578 /*******************************************************************************
579  *
580  * FUNCTION:    AcpiExDoLogicalOp
581  *
582  * PARAMETERS:  Opcode              - AML opcode
583  *              Operand0            - operand #0
584  *              Operand1            - operand #1
585  *              LogicalResult       - TRUE/FALSE result of the operation
586  *
587  * RETURN:      Status
588  *
589  * DESCRIPTION: Execute a logical AML opcode. The purpose of having all of the
590  *              functions here is to prevent a lot of pointer dereferencing
591  *              to obtain the operands and to simplify the generation of the
592  *              logical value. For the Numeric operators (LAnd and LOr), both
593  *              operands must be integers. For the other logical operators,
594  *              operands can be any combination of Integer/String/Buffer. The
595  *              first operand determines the type to which the second operand
596  *              will be converted.
597  *
598  *              Note: cleanest machine code seems to be produced by the code
599  *              below, rather than using statements of the form:
```

```
 *                  Result = (Operand0 == Operand1);
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExDoLogicalOp (
    UINT16                  Opcode,
    ACPI_OPERAND_OBJECT     *Operand0,
    ACPI_OPERAND_OBJECT     *Operand1,
    BOOLEAN                 *LogicalResult)
{
    ACPI_OPERAND_OBJECT     *LocalOperand1 = Operand1;
    UINT64                  Integer0;
    UINT64                  Integer1;
    UINT32                  Length0;
    UINT32                  Length1;
    ACPI_STATUS             Status = AE_OK;
    BOOLEAN                 LocalResult = FALSE;
    int                     Compare;


    ACPI_FUNCTION_TRACE (ExDoLogicalOp);


    /*
     * Convert the second operand if necessary. The first operand
     * determines the type of the second operand, (See the Data Types
     * section of the ACPI 3.0+ specification.)  Both object types are
     * guaranteed to be either Integer/String/Buffer by the operand
     * resolution mechanism.
     */
    switch (Operand0->Common.Type)
    {
    case ACPI_TYPE_INTEGER:

        Status = AcpiExConvertToInteger (Operand1, &LocalOperand1, 16);
        break;

    case ACPI_TYPE_STRING:

        Status = AcpiExConvertToString (Operand1, &LocalOperand1,
                    ACPI_IMPLICIT_CONVERT_HEX);
        break;

    case ACPI_TYPE_BUFFER:

        Status = AcpiExConvertToBuffer (Operand1, &LocalOperand1);
        break;

    default:

        Status = AE_AML_INTERNAL;
        break;
    }

    if (ACPI_FAILURE (Status))
    {
        goto Cleanup;
    }

    /*
     * Two cases: 1) Both Integers, 2) Both Strings or Buffers
     */
    if (Operand0->Common.Type == ACPI_TYPE_INTEGER)
    {
        /*
```



```
         * 1) Both operands are of type integer
         *    Note: LocalOperand1 may have changed above
         */
        Integer0 = Operand0->Integer.Value;
        Integer1 = LocalOperand1->Integer.Value;

        switch (Opcode)
        {
        case AML_LEQUAL_OP:             /* LEqual (Operand0, Operand1) */

            if (Integer0 == Integer1)
            {
                LocalResult = TRUE;
            }
            break;

        case AML_LGREATER_OP:           /* LGreater (Operand0, Operand1) */

            if (Integer0 > Integer1)
            {
                LocalResult = TRUE;
            }
            break;

        case AML_LLESS_OP:              /* LLess (Operand0, Operand1) */

            if (Integer0 < Integer1)
            {
                LocalResult = TRUE;
            }
            break;

        default:

            Status = AE_AML_INTERNAL;
            break;
        }
    }
    else
    {
        /*
         * 2) Both operands are Strings or both are Buffers
         *    Note: Code below takes advantage of common Buffer/String
         *          object fields. LocalOperand1 may have changed above. Use
         *          memcmp to handle nulls in buffers.
         */
        Length0 = Operand0->Buffer.Length;
        Length1 = LocalOperand1->Buffer.Length;

        /* Lexicographic compare: compare the data bytes */

        Compare = ACPI_MEMCMP (Operand0->Buffer.Pointer,
                    LocalOperand1->Buffer.Pointer,
                    (Length0 > Length1) ? Length1 : Length0);

        switch (Opcode)
        {
        case AML_LEQUAL_OP:             /* LEqual (Operand0, Operand1) */

            /* Length and all bytes must be equal */

            if ((Length0 == Length1) &&
                (Compare == 0))
            {
                /* Length and all bytes match ==> TRUE */
```

```
732                 LocalResult = TRUE;
733             }
734             break;

736         case AML_LGREATER_OP:           /* LGreater (Operand0, Operand1) */

738             if (Compare > 0)
739             {
740                 LocalResult = TRUE;
741                 goto Cleanup;   /* TRUE */
742             }
743             if (Compare < 0)
744             {
745                 goto Cleanup;   /* FALSE */
746             }

748             /* Bytes match (to shortest length), compare lengths */

750             if (Length0 > Length1)
751             {
752                 LocalResult = TRUE;
753             }
754             break;

756         case AML_LLESS_OP:              /* LLess (Operand0, Operand1) */

758             if (Compare > 0)
759             {
760                 goto Cleanup;   /* FALSE */
761             }
762             if (Compare < 0)
763             {
764                 LocalResult = TRUE;
765                 goto Cleanup;   /* TRUE */
766             }

768             /* Bytes match (to shortest length), compare lengths */

770             if (Length0 < Length1)
771             {
772                 LocalResult = TRUE;
773             }
774             break;

776         default:

778             Status = AE_AML_INTERNAL;
779             break;
780         }
781     }

783 Cleanup:

785     /* New object was created if implicit conversion performed - delete */

787     if (LocalOperand1 != Operand1)
788     {
789         AcpiUtRemoveReference (LocalOperand1);
790     }

792     /* Return the logical result and status */

794     *LogicalResult = LocalResult;
795     return_ACPI_STATUS (Status);
796 }
```

```
**********************************************************
   16339 Thu Dec 26 13:49:05 2013
new/usr/src/common/acpica/components/executer/exmutex.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exmutex - ASL Mutex Acquire/Release functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXMUTEX_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "acevents.h"

#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exmutex")

/* Local prototypes */

static void
AcpiExLinkMutex (
    ACPI_OPERAND_OBJECT     *ObjDesc,
```



```
    ACPI_THREAD_STATE       *Thread);


/*******************************************************************************
 *
 * FUNCTION:    AcpiExUnlinkMutex
 *
 * PARAMETERS:  ObjDesc             - The mutex to be unlinked
 *
 * RETURN:      None
 *
 * DESCRIPTION: Remove a mutex from the "AcquiredMutex" list
 *
 ******************************************************************************/

void
AcpiExUnlinkMutex (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
    ACPI_THREAD_STATE       *Thread = ObjDesc->Mutex.OwnerThread;


    if (!Thread)
    {
        return;
    }

    /* Doubly linked list */

    if (ObjDesc->Mutex.Next)
    {
        (ObjDesc->Mutex.Next)->Mutex.Prev = ObjDesc->Mutex.Prev;
    }

    if (ObjDesc->Mutex.Prev)
    {
        (ObjDesc->Mutex.Prev)->Mutex.Next = ObjDesc->Mutex.Next;

        /*
         * Migrate the previous sync level associated with this mutex to
         * the previous mutex on the list so that it may be preserved.
         * This handles the case where several mutexes have been acquired
         * at the same level, but are not released in opposite order.
         */
        (ObjDesc->Mutex.Prev)->Mutex.OriginalSyncLevel =
            ObjDesc->Mutex.OriginalSyncLevel;
    }
    else
    {
        Thread->AcquiredMutexList = ObjDesc->Mutex.Next;
    }
}
_____unchanged_portion_omitted_


/******************************************************************************
 *
 * FUNCTION:    AcpiExReleaseMutexObject
 *
 * PARAMETERS:  ObjDesc             - The object descriptor for this op
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Release a previously acquired Mutex, low level interface.
 *              Provides a common path that supports multiple releases (after
 *              previous multiple acquires) by the same thread.
```

```
 320  *
 321  * MUTEX:       Interpreter must be locked
 322  *
 323  * NOTE: This interface is called from three places:
 324  * 1) From AcpiExReleaseMutex, via an AML Acquire() operator
 325  * 2) From AcpiExReleaseGlobalLock when an AML Field access requires the
 326  *    global lock
 327  * 3) From the external interface, AcpiReleaseGlobalLock
 328  *
 329  ******************************************************************************/

 331 ACPI_STATUS
 332 AcpiExReleaseMutexObject (
 333     ACPI_OPERAND_OBJECT     *ObjDesc)
 334 {
 335     ACPI_STATUS             Status = AE_OK;


 338     ACPI_FUNCTION_TRACE (ExReleaseMutexObject);


 341     if (ObjDesc->Mutex.AcquisitionDepth == 0)
 342     {
 343         return_ACPI_STATUS (AE_NOT_ACQUIRED);
 344         return (AE_NOT_ACQUIRED);
 344     }

 346     /* Match multiple Acquires with multiple Releases */

 348     ObjDesc->Mutex.AcquisitionDepth--;
 349     if (ObjDesc->Mutex.AcquisitionDepth != 0)
 350     {
 351         /* Just decrement the depth and return */

 353         return_ACPI_STATUS (AE_OK);
 354     }

 356     if (ObjDesc->Mutex.OwnerThread)
 357     {
 358         /* Unlink the mutex from the owner's list */

 360         AcpiExUnlinkMutex (ObjDesc);
 361         ObjDesc->Mutex.OwnerThread = NULL;
 362     }

 364     /* Release the mutex, special case for Global Lock */

 366     if (ObjDesc == AcpiGbl_GlobalLockMutex)
 367     {
 368         Status = AcpiEvReleaseGlobalLock ();
 369     }
 370     else
 371     {
 372         AcpiOsReleaseMutex (ObjDesc->Mutex.OsMutex);
 373     }

 375     /* Clear mutex info */

 377     ObjDesc->Mutex.ThreadId = 0;
 378     return_ACPI_STATUS (Status);
 379 }
______unchanged_portion_omitted_


 491 /*******************************************************************************
 492  *
```



```
 493  * FUNCTION:    AcpiExReleaseAllMutexes
 494  *
 495  * PARAMETERS:  Thread              - Current executing thread object
 496  *
 497  * RETURN:      Status
 498  *
 499  * DESCRIPTION: Release all mutexes held by this thread
 500  *
 501  * NOTE: This function is called as the thread is exiting the interpreter.
 502  * Mutexes are not released when an individual control method is exited, but
 503  * only when the parent thread actually exits the interpreter. This allows one
 504  * method to acquire a mutex, and a different method to release it, as long as
 505  * this is performed underneath a single parent control method.
 506  *
 507  ******************************************************************************/

 509 void
 510 AcpiExReleaseAllMutexes (
 511     ACPI_THREAD_STATE       *Thread)
 512 {
 513     ACPI_OPERAND_OBJECT     *Next = Thread->AcquiredMutexList;
 514     ACPI_OPERAND_OBJECT     *ObjDesc;


 517     ACPI_FUNCTION_NAME (ExReleaseAllMutexes);
 518     ACPI_FUNCTION_ENTRY ();


 520     /* Traverse the list of owned mutexes, releasing each one */

 522     while (Next)
 523     {
 524         ObjDesc = Next;
 525         Next = ObjDesc->Mutex.Next;

 527         ObjDesc->Mutex.Prev = NULL;
 528         ObjDesc->Mutex.Next = NULL;
 529         ObjDesc->Mutex.AcquisitionDepth = 0;

 531         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
 532             "Force-releasing held mutex: %p\n", ObjDesc));

 534         /* Release the mutex, special case for Global Lock */

 536         if (ObjDesc == AcpiGbl_GlobalLockMutex)
 537         {
 538             /* Ignore errors */

 540             (void) AcpiEvReleaseGlobalLock ();
 541         }
 542         else
 543         {
 544             AcpiOsReleaseMutex (ObjDesc->Mutex.OsMutex);
 545         }

 547         /* Mark mutex unowned */

 549         ObjDesc->Mutex.OwnerThread = NULL;
 550         ObjDesc->Mutex.ThreadId = 0;

 552         /* Update Thread SyncLevel (Last mutex is the important one) */

 554         Thread->CurrentSyncLevel = ObjDesc->Mutex.OriginalSyncLevel;
 555     }
 556 }
______unchanged_portion_omitted_
```

****
13641 Thu Dec 26 13:49:05 2013
new/usr/src/common/acpica/components/executer/exnames.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exnames - interpreter/scanner name load/execute
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXNAMES_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exnames")

/* Local prototypes */

static char *
AcpiExAllocateNameString (
    UINT32                  PrefixCount,
    UINT32                  NumNameSegs);

static ACPI_STATUS
AcpiExNameSegment (
    UINT8                   **InAmlAddress,
    char                    *NameString);


/******************************************************************************
 *
 * FUNCTION:    AcpiExAllocateNameString
 *
 * PARAMETERS:  PrefixCount         - Count of parent levels. Special cases:
 *                                    (-1)==root,  0==none
 *              NumNameSegs         - count of 4-character name segments
 *
 * RETURN:      A pointer to the allocated string segment. This segment must
 *              be deleted by the caller.
 *
 * DESCRIPTION: Allocate a buffer for a name string. Ensure allocated name
 *              string is long enough, and set up prefix if any.
 *
 *****************************************************************************/

static char *
AcpiExAllocateNameString (
    UINT32                  PrefixCount,
    UINT32                  NumNameSegs)
{
    char                    *TempPtr;
    char                    *NameString;
    UINT32                  SizeNeeded;

    ACPI_FUNCTION_TRACE (ExAllocateNameString);


    /*
     * Allow room for all \ and ^ prefixes, all segments and a MultiNamePrefix.
     * Also, one byte for the null terminator.
     * This may actually be somewhat longer than needed.
     */
    if (PrefixCount == ACPI_UINT32_MAX)
    {
        /* Special case for root */

        SizeNeeded = 1 + (ACPI_NAME_SIZE * NumNameSegs) + 2 + 1;
    }
    else
    {
        SizeNeeded = PrefixCount + (ACPI_NAME_SIZE * NumNameSegs) + 2 + 1;
    }

    /*
     * Allocate a buffer for the name.
     * This buffer must be deleted by the caller!
     */
    NameString = ACPI_ALLOCATE (SizeNeeded);
    if (!NameString)
    {
        ACPI_ERROR ((AE_INFO,
            "Could not allocate size %u", SizeNeeded));
        return_PTR (NULL);
    }

    TempPtr = NameString;
```

```
125     /* Set up Root or Parent prefixes if needed */

127     if (PrefixCount == ACPI_UINT32_MAX)
128     {
129         *TempPtr++ = AML_ROOT_PREFIX;
130     }
131     else
132     {
133         while (PrefixCount--)
134         {
135             *TempPtr++ = AML_PARENT_PREFIX;
136         }
137     }


140     /* Set up Dual or Multi prefixes if needed */

142     if (NumNameSegs > 2)
143     {
144         /* Set up multi prefixes   */

146         *TempPtr++ = AML_MULTI_NAME_PREFIX_OP;
147         *TempPtr++ = (char) NumNameSegs;
148     }
149     else if (2 == NumNameSegs)
150     {
151         /* Set up dual prefixes */

153         *TempPtr++ = AML_DUAL_NAME_PREFIX;
154     }

156     /*
157      * Terminate string following prefixes. AcpiExNameSegment() will
158      * append the segment(s)
159      */
160     *TempPtr = 0;

162     return_PTR (NameString);
163 }
```

unchanged_portion_omitted_

```
264 /*****************************************************************************
265  *
266  * FUNCTION:    AcpiExGetNameString
267  *
268  * PARAMETERS:  DataType            - Object type to be associated with this
269  *                                    name
270  *              InAmlAddress        - Pointer to the namestring in the AML code
271  *              OutNameString       - Where the namestring is returned
272  *              OutNameLength       - Length of the returned string
273  *
274  * RETURN:      Status, namestring and length
275  *
276  * DESCRIPTION: Extract a full namepath from the AML byte stream,
277  *              including any prefixes.
278  *
279  ****************************************************************************/

281 ACPI_STATUS
282 AcpiExGetNameString (
283     ACPI_OBJECT_TYPE        DataType,
284     UINT8                   *InAmlAddress,
285     char                    **OutNameString,
286     UINT32                  *OutNameLength)
287 {
```



```
288     ACPI_STATUS             Status = AE_OK;
289     UINT8                   *AmlAddress = InAmlAddress;
290     char                    *NameString = NULL;
291     UINT32                  NumSegments;
292     UINT32                  PrefixCount = 0;
293     BOOLEAN                 HasPrefix = FALSE;


296     ACPI_FUNCTION_TRACE_PTR (ExGetNameString, AmlAddress);


299     if (ACPI_TYPE_LOCAL_REGION_FIELD == DataType   ||
300         ACPI_TYPE_LOCAL_BANK_FIELD == DataType     ||
301         ACPI_TYPE_LOCAL_INDEX_FIELD == DataType)
302     {
303         /* Disallow prefixes for types associated with FieldUnit names */

305         NameString = AcpiExAllocateNameString (0, 1);
306         if (!NameString)
307         {
308             Status = AE_NO_MEMORY;
309         }
310         else
311         {
312             Status = AcpiExNameSegment (&AmlAddress, NameString);
313         }
314     }
315     else
316     {
317         /*
318          * DataType is not a field name.
319          * Examine first character of name for root or parent prefix operators
320          */
321         switch (*AmlAddress)
322         {
323         case AML_ROOT_PREFIX:

325             ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "RootPrefix(\\) at %p\n",
326                 AmlAddress));

328             /*
329              * Remember that we have a RootPrefix --
330              * see comment in AcpiExAllocateNameString()
331              */
332             AmlAddress++;
333             PrefixCount = ACPI_UINT32_MAX;
334             HasPrefix = TRUE;
335             break;


337         case AML_PARENT_PREFIX:

339             /* Increment past possibly multiple parent prefixes */

341             do
342             {
343                 ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "ParentPrefix (^) at %p\n",
344                     AmlAddress));

346                 AmlAddress++;
347                 PrefixCount++;

349             } while (*AmlAddress == AML_PARENT_PREFIX);

351             HasPrefix = TRUE;
352             break;
```

```
354         default:
356             /* Not a prefix character */
358             break;
359         }
361         /* Examine first character of name for name segment prefix operator */
363         switch (*AmlAddress)
364         {
365         case AML_DUAL_NAME_PREFIX:
367             ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "DualNamePrefix at %p\n",
368                 AmlAddress));
370             AmlAddress++;
371             NameString = AcpiExAllocateNameString (PrefixCount, 2);
372             if (!NameString)
373             {
374                 Status = AE_NO_MEMORY;
375                 break;
376             }
378             /* Indicate that we processed a prefix */
380             HasPrefix = TRUE;
382             Status = AcpiExNameSegment (&AmlAddress, NameString);
383             if (ACPI_SUCCESS (Status))
384             {
385                 Status = AcpiExNameSegment (&AmlAddress, NameString);
386             }
387             break;
389         case AML_MULTI_NAME_PREFIX_OP:
391             ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "MultiNamePrefix at %p\n",
392                 AmlAddress));
394             /* Fetch count of segments remaining in name path */
396             AmlAddress++;
397             NumSegments = *AmlAddress;
399             NameString = AcpiExAllocateNameString (PrefixCount, NumSegments);
400             if (!NameString)
401             {
402                 Status = AE_NO_MEMORY;
403                 break;
404             }
406             /* Indicate that we processed a prefix */
408             AmlAddress++;
409             HasPrefix = TRUE;
411             while (NumSegments &&
412                    (Status = AcpiExNameSegment (&AmlAddress, NameString)) ==
413                        AE_OK)
414             {
415                 NumSegments--;
416             }
```



```
418             break;
420         case 0:
422             /* NullName valid as of 8-12-98 ASL/AML Grammar Update */
424             if (PrefixCount == ACPI_UINT32_MAX)
425             {
426                 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
427                     "NameSeg is \"\\\" followed by NULL\n"));
428             }
430             /* Consume the NULL byte */
432             AmlAddress++;
433             NameString = AcpiExAllocateNameString (PrefixCount, 0);
434             if (!NameString)
435             {
436                 Status = AE_NO_MEMORY;
437                 break;
438             }
440             break;
442         default:
444             /* Name segment string */
446             NameString = AcpiExAllocateNameString (PrefixCount, 1);
447             if (!NameString)
448             {
449                 Status = AE_NO_MEMORY;
450                 break;
451             }
453             Status = AcpiExNameSegment (&AmlAddress, NameString);
454             break;
455         }
456     }
458     if (AE_CTRL_PENDING == Status && HasPrefix)
459     {
460         /* Ran out of segments after processing a prefix */
462         ACPI_ERROR ((AE_INFO,
463             "Malformed Name at %p", NameString));
464         Status = AE_AML_BAD_NAME;
465     }
467     if (ACPI_FAILURE (Status))
468     {
469         if (NameString)
470         {
471             ACPI_FREE (NameString);
472         }
473         return_ACPI_STATUS (Status);
474     }
476     *OutNameString = NameString;
477     *OutNameLength = (UINT32) (AmlAddress - InAmlAddress);
479     return_ACPI_STATUS (Status);
480 }
```

```
**********************************************************
   34340 Thu Dec 26 13:49:05 2013
new/usr/src/common/acpica/components/executer/exoparg1.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exoparg1 - AML execution - opcodes with 1 argument
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXOPARG1_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acdispat.h"
#include "acinterp.h"
#include "amlcode.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exoparg1")
```



```
/*!
 * Naming convention for AML interpreter execution routines.
 *
 * The routines that begin execution of AML opcodes are named with a common
 * convention based upon the number of arguments, the number of target operands,
 * and whether or not a value is returned:
 *
 *      AcpiExOpcode_xA_yT_zR
 *
 * Where:
 *
 * xA - ARGUMENTS:    The number of arguments (input operands) that are
 *                    required for this opcode type (0 through 6 args).
 * yT - TARGETS:      The number of targets (output operands) that are required
 *                    for this opcode type (0, 1, or 2 targets).
 * zR - RETURN VALUE: Indicates whether this opcode type returns a value
 *                    as the function return (0 or 1).
 *
 * The AcpiExOpcode* functions are called via the Dispatcher component with
 * fully resolved operands.
!*/

/******************************************************************************
 *
 * FUNCTION:    AcpiExOpcode_0A_0T_1R
 *
 * PARAMETERS:  WalkState           - Current state (contains AML opcode)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute operator with no operands, one return value
 *
 *****************************************************************************/

ACPI_STATUS
AcpiExOpcode_0A_0T_1R (
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;


    ACPI_FUNCTION_TRACE_STR (ExOpcode_0A_0T_1R,
        AcpiPsGetOpcodeName (WalkState->Opcode));


    /* Examine the AML opcode */

    switch (WalkState->Opcode)
    {
    case AML_TIMER_OP:      /*  Timer () */

        /* Create a return object of type Integer */

        ReturnDesc = AcpiUtCreateIntegerObject (AcpiOsGetTimer ());
        if (!ReturnDesc)
        {
            Status = AE_NO_MEMORY;
            goto Cleanup;
        }
        break;

    default:                /*  Unknown opcode  */

        ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
            WalkState->Opcode));
```

```
125         Status = AE_AML_BAD_OPCODE;
126         break;
127     }

129 Cleanup:

131     /* Delete return object on error */

133     if ((ACPI_FAILURE (Status)) || WalkState->ResultObj)
134     {
135         AcpiUtRemoveReference (ReturnDesc);
136         WalkState->ResultObj = NULL;
137     }
138     else
139     {
140         /* Save the return value */

142         WalkState->ResultObj = ReturnDesc;
143     }

145     return_ACPI_STATUS (Status);
146 }


149 /*******************************************************************************
150  *
151  * FUNCTION:    AcpiExOpcode_1A_0T_0R
152  *
153  * PARAMETERS:  WalkState           - Current state (contains AML opcode)
154  *
155  * RETURN:      Status
156  *
157  * DESCRIPTION: Execute Type 1 monadic operator with numeric operand on
158  *              object stack
159  *
160  ******************************************************************************/

162 ACPI_STATUS
163 AcpiExOpcode_1A_0T_0R (
164     ACPI_WALK_STATE         *WalkState)
165 {
166     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
167     ACPI_STATUS             Status = AE_OK;


170     ACPI_FUNCTION_TRACE_STR (ExOpcode_1A_0T_0R,
171         AcpiPsGetOpcodeName (WalkState->Opcode));


174     /* Examine the AML opcode */

176     switch (WalkState->Opcode)
177     {
178     case AML_RELEASE_OP:    /*  Release (MutexObject) */

180         Status = AcpiExReleaseMutex (Operand[0], WalkState);
181         break;


183     case AML_RESET_OP:      /*  Reset (EventObject) */

185         Status = AcpiExSystemResetEvent (Operand[0]);
186         break;


188     case AML_SIGNAL_OP:     /*  Signal (EventObject) */
```



```
190         Status = AcpiExSystemSignalEvent (Operand[0]);
191         break;


193     case AML_SLEEP_OP:      /*  Sleep (MsecTime) */

195         Status = AcpiExSystemDoSleep (Operand[0]->Integer.Value);
196         break;


198     case AML_STALL_OP:      /*  Stall (UsecTime) */

200         Status = AcpiExSystemDoStall ((UINT32) Operand[0]->Integer.Value);
201         break;


203     case AML_UNLOAD_OP:     /*  Unload (Handle) */

205         Status = AcpiExUnloadTable (Operand[0]);
206         break;


208     default:                /*  Unknown opcode  */

210         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
211             WalkState->Opcode));
212         Status = AE_AML_BAD_OPCODE;
213         break;
214     }

216     return_ACPI_STATUS (Status);
217 }
```

unchanged_portion_omitted_

```
269 /*******************************************************************************
270  *
271  * FUNCTION:    AcpiExOpcode_1A_1T_1R
272  *
273  * PARAMETERS:  WalkState           - Current state (contains AML opcode)
274  *
275  * RETURN:      Status
276  *
277  * DESCRIPTION: Execute opcode with one argument, one target, and a
278  *              return value.
279  *
280  ******************************************************************************/

282 ACPI_STATUS
283 AcpiExOpcode_1A_1T_1R (
284     ACPI_WALK_STATE         *WalkState)
285 {
286     ACPI_STATUS             Status = AE_OK;
287     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
288     ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
289     ACPI_OPERAND_OBJECT     *ReturnDesc2 = NULL;
290     UINT32                  Temp32;
291     UINT32                  i;
292     UINT64                  PowerOfTen;
293     UINT64                  Digit;


296     ACPI_FUNCTION_TRACE_STR (ExOpcode_1A_1T_1R,
297         AcpiPsGetOpcodeName (WalkState->Opcode));
```

```
300     /* Examine the AML opcode */

302     switch (WalkState->Opcode)
303     {
304     case AML_BIT_NOT_OP:
305     case AML_FIND_SET_LEFT_BIT_OP:
306     case AML_FIND_SET_RIGHT_BIT_OP:
307     case AML_FROM_BCD_OP:
308     case AML_TO_BCD_OP:
309     case AML_COND_REF_OF_OP:

311         /* Create a return object of type Integer for these opcodes */

313         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
314         if (!ReturnDesc)
315         {
316             Status = AE_NO_MEMORY;
317             goto Cleanup;
318         }

320         switch (WalkState->Opcode)
321         {
322         case AML_BIT_NOT_OP:            /* Not (Operand, Result)  */

324             ReturnDesc->Integer.Value = ~Operand[0]->Integer.Value;
325             break;


327         case AML_FIND_SET_LEFT_BIT_OP:  /* FindSetLeftBit (Operand, Result) */

329             ReturnDesc->Integer.Value = Operand[0]->Integer.Value;

331             /*
332              * Acpi specification describes Integer type as a little
333              * endian unsigned value, so this boundary condition is valid.
334              */
335             for (Temp32 = 0; ReturnDesc->Integer.Value &&
336                              Temp32 < ACPI_INTEGER_BIT_SIZE; ++Temp32)
337             {
338                 ReturnDesc->Integer.Value >>= 1;
339             }

341             ReturnDesc->Integer.Value = Temp32;
342             break;


344         case AML_FIND_SET_RIGHT_BIT_OP: /* FindSetRightBit (Operand, Result) */

346             ReturnDesc->Integer.Value = Operand[0]->Integer.Value;

348             /*
349              * The Acpi specification describes Integer type as a little
350              * endian unsigned value, so this boundary condition is valid.
351              */
352             for (Temp32 = 0; ReturnDesc->Integer.Value &&
353                              Temp32 < ACPI_INTEGER_BIT_SIZE; ++Temp32)
354             {
355                 ReturnDesc->Integer.Value <<= 1;
356             }

358             /* Since the bit position is one-based, subtract from 33 (65) */

360             ReturnDesc->Integer.Value =
361                 Temp32 == 0 ? 0 : (ACPI_INTEGER_BIT_SIZE + 1) - Temp32;
362             break;
```



```
364         case AML_FROM_BCD_OP:           /* FromBcd (BCDValue, Result)  */

365             /*
366              * The 64-bit ACPI integer can hold 16 4-bit BCD characters
367              * (if table is 32-bit, integer can hold 8 BCD characters)
368              * Convert each 4-bit BCD value
369              */
370             PowerOfTen = 1;
371             ReturnDesc->Integer.Value = 0;
372             Digit = Operand[0]->Integer.Value;

374             /* Convert each BCD digit (each is one nybble wide) */

376             for (i = 0; (i < AcpiGbl_IntegerNybbleWidth) && (Digit > 0); i++)
377             {
378                 /* Get the least significant 4-bit BCD digit */

380                 Temp32 = ((UINT32) Digit) & 0xF;

382                 /* Check the range of the digit */

384                 if (Temp32 > 9)
385                 {
386                     ACPI_ERROR ((AE_INFO,
387                         "BCD digit too large (not decimal): 0x%X",
388                         Temp32));

390                     Status = AE_AML_NUMERIC_OVERFLOW;
391                     goto Cleanup;
392                 }

394                 /* Sum the digit into the result with the current power of 10 */

396                 ReturnDesc->Integer.Value +=
397                     (((UINT64) Temp32) * PowerOfTen);

399                 /* Shift to next BCD digit */

401                 Digit >>= 4;

403                 /* Next power of 10 */

405                 PowerOfTen *= 10;
406             }
407             break;


409         case AML_TO_BCD_OP:             /* ToBcd (Operand, Result)  */

411             ReturnDesc->Integer.Value = 0;
412             Digit = Operand[0]->Integer.Value;

414             /* Each BCD digit is one nybble wide */

416             for (i = 0; (i < AcpiGbl_IntegerNybbleWidth) && (Digit > 0); i++)
417             {
418                 (void) AcpiUtShortDivide (Digit, 10, &Digit, &Temp32);

420                 /*
421                  * Insert the BCD digit that resides in the
422                  * remainder from above
423                  */
424                 ReturnDesc->Integer.Value |=
425                     (((UINT64) Temp32) << ACPI_MUL_4 (i));
```

```
426             }
428             /* Overflow if there is any data left in Digit */
430             if (Digit > 0)
431             {
432                 ACPI_ERROR ((AE_INFO,
433                     "Integer too large to convert to BCD: 0x%8.8X%8.8X",
434                     ACPI_FORMAT_UINT64 (Operand[0]->Integer.Value)));
435                 Status = AE_AML_NUMERIC_OVERFLOW;
436                 goto Cleanup;
437             }
438             break;
440         case AML_COND_REF_OF_OP:        /* CondRefOf (SourceObject, Result)  */
441             /*
442              * This op is a little strange because the internal return value is
443              * different than the return value stored in the result descriptor
444              * (There are really two return values)
445              */
446             if ((ACPI_NAMESPACE_NODE *) Operand[0] == AcpiGbl_RootNode)
447             {
448                 /*
449                  * This means that the object does not exist in the namespace,
450                  * return FALSE
451                  */
452                 ReturnDesc->Integer.Value = 0;
453                 goto Cleanup;
454             }
456             /* Get the object reference, store it, and remove our reference */
458             Status = AcpiExGetObjectReference (Operand[0],
459                         &ReturnDesc2, WalkState);
460             if (ACPI_FAILURE (Status))
461             {
462                 goto Cleanup;
463             }
465             Status = AcpiExStore (ReturnDesc2, Operand[1], WalkState);
466             AcpiUtRemoveReference (ReturnDesc2);
468             /* The object exists in the namespace, return TRUE */
470             ReturnDesc->Integer.Value = ACPI_UINT64_MAX;
471             goto Cleanup;
474         default:
476             /* No other opcodes get here */
478             break;
479         }
480         break;
482     case AML_STORE_OP:              /* Store (Source, Target) */
483         /*
484          * A store operand is typically a number, string, buffer or lvalue
485          * Be careful about deleting the source object,
486          * since the object itself may have been stored.
487          */
```



```
488         Status = AcpiExStore (Operand[0], Operand[1], WalkState);
489         if (ACPI_FAILURE (Status))
490         {
491             return_ACPI_STATUS (Status);
492         }
494         /* It is possible that the Store already produced a return object */
496         if (!WalkState->ResultObj)
497         {
498             /*
499              * Normally, we would remove a reference on the Operand[0]
500              * parameter; But since it is being used as the internal return
501              * object (meaning we would normally increment it), the two
502              * cancel out, and we simply don't do anything.
503              */
504             WalkState->ResultObj = Operand[0];
505             WalkState->Operands[0] = NULL;  /* Prevent deletion */
506         }
507         return_ACPI_STATUS (Status);
509     /*
510      * ACPI 2.0 Opcodes
511      */
512     case AML_COPY_OP:               /* Copy (Source, Target) */
514         Status = AcpiUtCopyIobjectToIobject (Operand[0], &ReturnDesc,
515                     WalkState);
516         break;
518     case AML_TO_DECSTRING_OP:       /* ToDecimalString (Data, Result) */
520         Status = AcpiExConvertToString (Operand[0], &ReturnDesc,
521                     ACPI_EXPLICIT_CONVERT_DECIMAL);
522         if (ReturnDesc == Operand[0])
523         {
524             /* No conversion performed, add ref to handle return value */
525             AcpiUtAddReference (ReturnDesc);
526         }
527         break;
529     case AML_TO_HEXSTRING_OP:       /* ToHexString (Data, Result) */
531         Status = AcpiExConvertToString (Operand[0], &ReturnDesc,
532                     ACPI_EXPLICIT_CONVERT_HEX);
533         if (ReturnDesc == Operand[0])
534         {
535             /* No conversion performed, add ref to handle return value */
536             AcpiUtAddReference (ReturnDesc);
537         }
538         break;
540     case AML_TO_BUFFER_OP:          /* ToBuffer (Data, Result) */
542         Status = AcpiExConvertToBuffer (Operand[0], &ReturnDesc);
543         if (ReturnDesc == Operand[0])
544         {
545             /* No conversion performed, add ref to handle return value */
546             AcpiUtAddReference (ReturnDesc);
547         }
548         break;
```

```
550     case AML_TO_INTEGER_OP:         /* ToInteger (Data, Result) */

552         Status = AcpiExConvertToInteger (Operand[0], &ReturnDesc,
553                     ACPI_ANY_BASE);
554         if (ReturnDesc == Operand[0])
555         {
556             /* No conversion performed, add ref to handle return value */
557             AcpiUtAddReference (ReturnDesc);
558         }
559         break;


561     case AML_SHIFT_LEFT_BIT_OP:     /* ShiftLeftBit (Source, BitNum)  */
562     case AML_SHIFT_RIGHT_BIT_OP:    /* ShiftRightBit (Source, BitNum) */

564         /* These are two obsolete opcodes */

566         ACPI_ERROR ((AE_INFO,
567             "%s is obsolete and not implemented",
568             AcpiPsGetOpcodeName (WalkState->Opcode)));
569         Status = AE_SUPPORT;
570         goto Cleanup;


572     default:                        /* Unknown opcode */

574         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
575             WalkState->Opcode));
576         Status = AE_AML_BAD_OPCODE;
577         goto Cleanup;
578     }

580     if (ACPI_SUCCESS (Status))
581     {
582         /* Store the return value computed above into the target object */

584         Status = AcpiExStore (ReturnDesc, Operand[1], WalkState);
585     }


588 Cleanup:

590     /* Delete return object on error */

592     if (ACPI_FAILURE (Status))
593     {
594         AcpiUtRemoveReference (ReturnDesc);
595     }

597     /* Save return object on success */

599     else if (!WalkState->ResultObj)
600     {
601         WalkState->ResultObj = ReturnDesc;
602     }

604     return_ACPI_STATUS (Status);
605 }


608 /*****************************************************************************
609  *
610  * FUNCTION:    AcpiExOpcode_1A_0T_1R
611  *
612  * PARAMETERS:  WalkState           - Current state (contains AML opcode)
```



```
613  *
614  * RETURN:      Status
615  *
616  * DESCRIPTION: Execute opcode with one argument, no target, and a return value
617  *
618  ****************************************************************************/

620 ACPI_STATUS
621 AcpiExOpcode_1A_0T_1R (
622     ACPI_WALK_STATE         *WalkState)
623 {
624     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
625     ACPI_OPERAND_OBJECT     *TempDesc;
626     ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
627     ACPI_STATUS             Status = AE_OK;
628     UINT32                  Type;
629     UINT64                  Value;


632     ACPI_FUNCTION_TRACE_STR (ExOpcode_1A_0T_1R,
633         AcpiPsGetOpcodeName (WalkState->Opcode));


636     /* Examine the AML opcode */

638     switch (WalkState->Opcode)
639     {
640     case AML_LNOT_OP:               /* LNot (Operand) */

642         ReturnDesc = AcpiUtCreateIntegerObject ((UINT64) 0);
643         if (!ReturnDesc)
644         {
645             Status = AE_NO_MEMORY;
646             goto Cleanup;
647         }

649         /*
650          * Set result to ONES (TRUE) if Value == 0. Note:
651          * ReturnDesc->Integer.Value is initially == 0 (FALSE) from above.
652          */
653         if (!Operand[0]->Integer.Value)
654         {
655             ReturnDesc->Integer.Value = ACPI_UINT64_MAX;
656         }
657         break;


659     case AML_DECREMENT_OP:          /* Decrement (Operand)  */
660     case AML_INCREMENT_OP:          /* Increment (Operand)  */

661         /*
662          * Create a new integer. Can't just get the base integer and
663          * increment it because it may be an Arg or Field.
664          */
665         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
666         if (!ReturnDesc)
667         {
668             Status = AE_NO_MEMORY;
669             goto Cleanup;
670         }

672         /*
673          * Since we are expecting a Reference operand, it can be either a
674          * NS Node or an internal object.
675          */
676         TempDesc = Operand[0];
```

```
677         if (ACPI_GET_DESCRIPTOR_TYPE (TempDesc) == ACPI_DESC_TYPE_OPERAND)
678         {
679             /* Internal reference object - prevent deletion */

681             AcpiUtAddReference (TempDesc);
682         }

684         /*
685          * Convert the Reference operand to an Integer (This removes a
686          * reference on the Operand[0] object)
687          *
688          * NOTE:  We use LNOT_OP here in order to force resolution of the
689          * reference operand to an actual integer.
690          */
691         Status = AcpiExResolveOperands (AML_LNOT_OP, &TempDesc, WalkState);
692         if (ACPI_FAILURE (Status))
693         {
694             ACPI_EXCEPTION ((AE_INFO, Status,
695                 "While resolving operands for [%s]",
696                 AcpiPsGetOpcodeName (WalkState->Opcode)));

698             goto Cleanup;
699         }

701         /*
702          * TempDesc is now guaranteed to be an Integer object --
703          * Perform the actual increment or decrement
704          */
705         if (WalkState->Opcode == AML_INCREMENT_OP)
706         {
707             ReturnDesc->Integer.Value = TempDesc->Integer.Value +1;
708         }
709         else
710         {
711             ReturnDesc->Integer.Value = TempDesc->Integer.Value -1;
712         }

714         /* Finished with this Integer object */

716         AcpiUtRemoveReference (TempDesc);

718         /*
719          * Store the result back (indirectly) through the original
720          * Reference object
721          */
722         Status = AcpiExStore (ReturnDesc, Operand[0], WalkState);
723         break;


725     case AML_TYPE_OP:               /* ObjectType (SourceObject) */

726         /*
727          * Note: The operand is not resolved at this point because we want to
728          * get the associated object, not its value. For example, we don't
729          * want to resolve a FieldUnit to its value, we want the actual
730          * FieldUnit object.
731          */

733         /* Get the type of the base object */

735         Status = AcpiExResolveMultiple (WalkState, Operand[0], &Type, NULL);
736         if (ACPI_FAILURE (Status))
737         {
738             goto Cleanup;
739         }
```



```
741         /* Allocate a descriptor to hold the type. */

743         ReturnDesc = AcpiUtCreateIntegerObject ((UINT64) Type);
744         if (!ReturnDesc)
745         {
746             Status = AE_NO_MEMORY;
747             goto Cleanup;
748         }
749         break;


751     case AML_SIZE_OF_OP:            /* SizeOf (SourceObject)  */

752         /*
753          * Note: The operand is not resolved at this point because we want to
754          * get the associated object, not its value.
755          */

757         /* Get the base object */

759         Status = AcpiExResolveMultiple (WalkState,
760                     Operand[0], &Type, &TempDesc);
761         if (ACPI_FAILURE (Status))
762         {
763             goto Cleanup;
764         }

766         /*
767          * The type of the base object must be integer, buffer, string, or
768          * package. All others are not supported.
769          *
770          * NOTE: Integer is not specifically supported by the ACPI spec,
771          * but is supported implicitly via implicit operand conversion.
772          * rather than bother with conversion, we just use the byte width
773          * global (4 or 8 bytes).
774          */
775         switch (Type)
776         {
777         case ACPI_TYPE_INTEGER:

779             Value = AcpiGbl_IntegerByteWidth;
780             break;

782         case ACPI_TYPE_STRING:

784             Value = TempDesc->String.Length;
785             break;

787         case ACPI_TYPE_BUFFER:

789             /* Buffer arguments may not be evaluated at this point */

791             Status = AcpiDsGetBufferArguments (TempDesc);
792             Value = TempDesc->Buffer.Length;
793             break;

795         case ACPI_TYPE_PACKAGE:

797             /* Package arguments may not be evaluated at this point */

799             Status = AcpiDsGetPackageArguments (TempDesc);
800             Value = TempDesc->Package.Count;
801             break;

803         default:
```

```
805             ACPI_ERROR ((AE_INFO,
806                 "Operand must be Buffer/Integer/String/Package - found type %s",
807                 AcpiUtGetTypeName (Type)));
808             Status = AE_AML_OPERAND_TYPE;
809             goto Cleanup;
810         }

812         if (ACPI_FAILURE (Status))
813         {
814             goto Cleanup;
815         }

817         /*
818          * Now that we have the size of the object, create a result
819          * object to hold the value
820          */
821         ReturnDesc = AcpiUtCreateIntegerObject (Value);
822         if (!ReturnDesc)
823         {
824             Status = AE_NO_MEMORY;
825             goto Cleanup;
826         }
827         break;


830     case AML_REF_OF_OP:             /* RefOf (SourceObject) */

832         Status = AcpiExGetObjectReference (Operand[0], &ReturnDesc, WalkState);
833         if (ACPI_FAILURE (Status))
834         {
835             goto Cleanup;
836         }
837         break;


840     case AML_DEREF_OF_OP:           /* DerefOf (ObjReference | String) */

842         /* Check for a method local or argument, or standalone String */

844         if (ACPI_GET_DESCRIPTOR_TYPE (Operand[0]) == ACPI_DESC_TYPE_NAMED)
845         {
846             TempDesc = AcpiNsGetAttachedObject (
847                         (ACPI_NAMESPACE_NODE *) Operand[0]);
848             if (TempDesc &&
849                  ((TempDesc->Common.Type == ACPI_TYPE_STRING) ||
850                   (TempDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE)))
851             {
852                 Operand[0] = TempDesc;
853                 AcpiUtAddReference (TempDesc);
854             }
855             else
856             {
857                 Status = AE_AML_OPERAND_TYPE;
858                 goto Cleanup;
859             }
860         }
861         else
862         {
863             switch ((Operand[0])->Common.Type)
864             {
865             case ACPI_TYPE_LOCAL_REFERENCE:
866                 /*
867                  * This is a DerefOf (LocalX | ArgX)
868                  *
869                  * Must resolve/dereference the local/arg reference first
870                  */
```



```
871                 switch (Operand[0]->Reference.Class)
872                 {
873                 case ACPI_REFCLASS_LOCAL:
874                 case ACPI_REFCLASS_ARG:

876                     /* Set Operand[0] to the value of the local/arg */

878                     Status = AcpiDsMethodDataGetValue (
879                                 Operand[0]->Reference.Class,
880                                 Operand[0]->Reference.Value,
881                                 WalkState, &TempDesc);
882                     if (ACPI_FAILURE (Status))
883                     {
884                         goto Cleanup;
885                     }

887                     /*
888                      * Delete our reference to the input object and
889                      * point to the object just retrieved
890                      */
891                     AcpiUtRemoveReference (Operand[0]);
892                     Operand[0] = TempDesc;
893                     break;

895                 case ACPI_REFCLASS_REFOF:

897                     /* Get the object to which the reference refers */

899                     TempDesc = Operand[0]->Reference.Object;
900                     AcpiUtRemoveReference (Operand[0]);
901                     Operand[0] = TempDesc;
902                     break;

904                 default:

906                     /* Must be an Index op - handled below */
907                     break;
908                 }
909                 break;

911             case ACPI_TYPE_STRING:

913                 break;

915             default:

917                 Status = AE_AML_OPERAND_TYPE;
918                 goto Cleanup;
919             }
920         }

922         if (ACPI_GET_DESCRIPTOR_TYPE (Operand[0]) != ACPI_DESC_TYPE_NAMED)
923         {
924             if ((Operand[0])->Common.Type == ACPI_TYPE_STRING)
925             {
926                 /*
927                  * This is a DerefOf (String). The string is a reference
928                  * to a named ACPI object.
929                  *
930                  * 1) Find the owning Node
931                  * 2) Dereference the node to an actual object. Could be a
932                  *    Field, so we need to resolve the node to a value.
933                  */
934                 Status = AcpiNsGetNode (WalkState->ScopeInfo->Scope.Node,
935                             Operand[0]->String.Pointer,
936                             ACPI_NS_SEARCH_PARENT,
```

```
                                ACPI_CAST_INDIRECT_PTR (
                                    ACPI_NAMESPACE_NODE, &ReturnDesc));
                if (ACPI_FAILURE (Status))
                {
                    goto Cleanup;
                }

                Status = AcpiExResolveNodeToValue (
                                ACPI_CAST_INDIRECT_PTR (
                                    ACPI_NAMESPACE_NODE, &ReturnDesc),
                                WalkState);
                goto Cleanup;
            }
        }

        /* Operand[0] may have changed from the code above */

        if (ACPI_GET_DESCRIPTOR_TYPE (Operand[0]) == ACPI_DESC_TYPE_NAMED)
        {
            /*
             * This is a DerefOf (ObjectReference)
             * Get the actual object from the Node (This is the dereference).
             * This case may only happen when a LocalX or ArgX is
             * dereferenced above.
             */
            ReturnDesc = AcpiNsGetAttachedObject (
                            (ACPI_NAMESPACE_NODE *) Operand[0]);
            AcpiUtAddReference (ReturnDesc);
        }
        else
        {
            /*
             * This must be a reference object produced by either the
             * Index() or RefOf() operator
             */
            switch (Operand[0]->Reference.Class)
            {
            case ACPI_REFCLASS_INDEX:

                /*
                 * The target type for the Index operator must be
                 * either a Buffer or a Package
                 */
                switch (Operand[0]->Reference.TargetType)
                {
                case ACPI_TYPE_BUFFER_FIELD:

                    TempDesc = Operand[0]->Reference.Object;

                    /*
                     * Create a new object that contains one element of the
                     * buffer -- the element pointed to by the index.
                     *
                     * NOTE: index into a buffer is NOT a pointer to a
                     * sub-buffer of the main buffer, it is only a pointer to a
                     * single element (byte) of the buffer!
                     *
                     * Since we are returning the value of the buffer at the
                     * indexed location, we don't need to add an additional
                     * reference to the buffer itself.
                     */
                    ReturnDesc = AcpiUtCreateIntegerObject ((UINT64)
                        TempDesc->Buffer.Pointer[Operand[0]->Reference.Value]);
                    if (!ReturnDesc)
                    {
                        Status = AE_NO_MEMORY;
```



```
                        goto Cleanup;
                    }
                    break;


                case ACPI_TYPE_PACKAGE:

                    /*
                     * Return the referenced element of the package. We must
                     * add another reference to the referenced object, however.
                     */
                    ReturnDesc = *(Operand[0]->Reference.Where);
                    if (!ReturnDesc)
                    if (ReturnDesc)
                    {
                        /*
                         * Element is NULL, do not allow the dereference.
                         * This provides compatibility with other ACPI
                         * implementations.
                         */
                        return_ACPI_STATUS (AE_AML_UNINITIALIZED_ELEMENT);
                    }

                    AcpiUtAddReference (ReturnDesc);
                    }
                    break;


                default:

                    ACPI_ERROR ((AE_INFO,
                        "Unknown Index TargetType 0x%X in reference object %p",
                        Operand[0]->Reference.TargetType, Operand[0]));
                    Status = AE_AML_OPERAND_TYPE;
                    goto Cleanup;
                }
                break;


            case ACPI_REFCLASS_REFOF:

                ReturnDesc = Operand[0]->Reference.Object;

                if (ACPI_GET_DESCRIPTOR_TYPE (ReturnDesc) ==
                    ACPI_DESC_TYPE_NAMED)
                {
                    ReturnDesc = AcpiNsGetAttachedObject (
                        (ACPI_NAMESPACE_NODE *) ReturnDesc);
                    if (!ReturnDesc)
                    {
                        break;
                    }

                   /*
                    * June 2013:
                    * BufferFields/FieldUnits require additional resolution
                    */
                    switch (ReturnDesc->Common.Type)
                    {
                    case ACPI_TYPE_BUFFER_FIELD:
                    case ACPI_TYPE_LOCAL_REGION_FIELD:
                    case ACPI_TYPE_LOCAL_BANK_FIELD:
                    case ACPI_TYPE_LOCAL_INDEX_FIELD:
                    /* Add another reference to the object! */

                        Status = AcpiExReadDataFromField (WalkState,
```

```
                                ReturnDesc, &TempDesc);
                            if (ACPI_FAILURE (Status))
                            {
                                goto Cleanup;
                            }

                            ReturnDesc = TempDesc;
                            break;

                        default:

                            /* Add another reference to the object */

                            AcpiUtAddReference (ReturnDesc);
                            break;
                        }
                    }
                    break;

                default:

                default:
                    ACPI_ERROR ((AE_INFO,
                        "Unknown class in reference(%p) - 0x%2.2X",
                        Operand[0], Operand[0]->Reference.Class));

                    Status = AE_TYPE;
                    goto Cleanup;
                }
            }
            break;


        default:

            ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
                WalkState->Opcode));
            Status = AE_AML_BAD_OPCODE;
            goto Cleanup;
        }


    Cleanup:

        /* Delete return object on error */

        if (ACPI_FAILURE (Status))
        {
            AcpiUtRemoveReference (ReturnDesc);
        }

        /* Save return object on success */

        else
        {
            WalkState->ResultObj = ReturnDesc;
        }

        return_ACPI_STATUS (Status);
    }
```

****
17928 Thu Dec 26 13:49:06 2013
new/usr/src/common/acpica/components/executer/exoparg2.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exoparg2 - AML execution - opcodes with 2 arguments
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __EXOPARG2_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acinterp.h"
#include "acevents.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exoparg2")


/*!
```



```
 * Naming convention for AML interpreter execution routines.
 *
 * The routines that begin execution of AML opcodes are named with a common
 * convention based upon the number of arguments, the number of target operands,
 * and whether or not a value is returned:
 *
 *      AcpiExOpcode_xA_yT_zR
 *
 * Where:
 *
 * xA - ARGUMENTS:    The number of arguments (input operands) that are
 *                    required for this opcode type (1 through 6 args).
 * yT - TARGETS:      The number of targets (output operands) that are required
 *                    for this opcode type (0, 1, or 2 targets).
 * zR - RETURN VALUE: Indicates whether this opcode type returns a value
 *                    as the function return (0 or 1).
 *
 * The AcpiExOpcode* functions are called via the Dispatcher component with
 * fully resolved operands.
!*/


/******************************************************************************
 *
 * FUNCTION:    AcpiExOpcode_2A_0T_0R
 *
 * PARAMETERS:  WalkState           - Current walk state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute opcode with two arguments, no target, and no return
 *              value.
 *
 * ALLOCATION:  Deletes both operands
 *
 *****************************************************************************/

ACPI_STATUS
AcpiExOpcode_2A_0T_0R (
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
    ACPI_NAMESPACE_NODE     *Node;
    UINT32                  Value;
    ACPI_STATUS             Status = AE_OK;


    ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_0T_0R,
            AcpiPsGetOpcodeName (WalkState->Opcode));


    /* Examine the opcode */

    switch (WalkState->Opcode)
    {
    case AML_NOTIFY_OP:         /* Notify (NotifyObject, NotifyValue) */

        /* The first operand is a namespace node */

        Node = (ACPI_NAMESPACE_NODE *) Operand[0];

        /* Second value is the notify value */

        Value = (UINT32) Operand[1]->Integer.Value;

        /* Are notifies allowed on this object? */
```

```
127         if (!AcpiEvIsNotifyObject (Node))
128         {
129             ACPI_ERROR ((AE_INFO,
130                 "Unexpected notify object type [%s]",
131                 AcpiUtGetTypeName (Node->Type)));

133             Status = AE_AML_OPERAND_TYPE;
134             break;
135         }

137         /*
138          * Dispatch the notify to the appropriate handler
139          * NOTE: the request is queued for execution after this method
140          * completes. The notify handlers are NOT invoked synchronously
141          * from this thread -- because handlers may in turn run other
142          * control methods.
143          */
144         Status = AcpiEvQueueNotifyRequest (Node, Value);
145         break;


147     default:

149         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
150             WalkState->Opcode));
151         Status = AE_AML_BAD_OPCODE;
152     }

154     return_ACPI_STATUS (Status);
155 }


158 /*******************************************************************************
159  *
160  * FUNCTION:    AcpiExOpcode_2A_2T_1R
161  *
162  * PARAMETERS:  WalkState           - Current walk state
163  *
164  * RETURN:      Status
165  *
166  * DESCRIPTION: Execute a dyadic operator (2 operands) with 2 output targets
167  *              and one implicit return value.
168  *
169  ******************************************************************************/

171 ACPI_STATUS
172 AcpiExOpcode_2A_2T_1R (
173     ACPI_WALK_STATE         *WalkState)
174 {
175     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
176     ACPI_OPERAND_OBJECT     *ReturnDesc1 = NULL;
177     ACPI_OPERAND_OBJECT     *ReturnDesc2 = NULL;
178     ACPI_STATUS             Status;


181     ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_2T_1R,
182         AcpiPsGetOpcodeName (WalkState->Opcode));


185     /* Execute the opcode */

187     switch (WalkState->Opcode)
188     {
189     case AML_DIVIDE_OP:
```



```
191         /* Divide (Dividend, Divisor, RemainderResult QuotientResult) */

193         ReturnDesc1 = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
194         if (!ReturnDesc1)
195         {
196             Status = AE_NO_MEMORY;
197             goto Cleanup;
198         }

200         ReturnDesc2 = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
201         if (!ReturnDesc2)
202         {
203             Status = AE_NO_MEMORY;
204             goto Cleanup;
205         }

207         /* Quotient to ReturnDesc1, remainder to ReturnDesc2 */

209         Status = AcpiUtDivide (Operand[0]->Integer.Value,
210                                Operand[1]->Integer.Value,
211                                &ReturnDesc1->Integer.Value,
212                                &ReturnDesc2->Integer.Value);
213         if (ACPI_FAILURE (Status))
214         {
215             goto Cleanup;
216         }
217         break;


219     default:

221         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
222             WalkState->Opcode));
223         Status = AE_AML_BAD_OPCODE;
224         goto Cleanup;
225     }

227     /* Store the results to the target reference operands */

229     Status = AcpiExStore (ReturnDesc2, Operand[2], WalkState);
230     if (ACPI_FAILURE (Status))
231     {
232         goto Cleanup;
233     }

235     Status = AcpiExStore (ReturnDesc1, Operand[3], WalkState);
236     if (ACPI_FAILURE (Status))
237     {
238         goto Cleanup;
239     }

241 Cleanup:
242     /*
243      * Since the remainder is not returned indirectly, remove a reference to
244      * it. Only the quotient is returned indirectly.
245      */
246     AcpiUtRemoveReference (ReturnDesc2);

248     if (ACPI_FAILURE (Status))
249     {
250         /* Delete the return object */

252         AcpiUtRemoveReference (ReturnDesc1);
253     }

255     /* Save return object (the remainder) on success */
```

```
257     else
258     {
259         WalkState->ResultObj = ReturnDesc1;
260     }

262     return_ACPI_STATUS (Status);
263 }


266 /*****************************************************************************
267  *
268  * FUNCTION:    AcpiExOpcode_2A_1T_1R
269  *
270  * PARAMETERS:  WalkState           - Current walk state
271  *
272  * RETURN:      Status
273  *
274  * DESCRIPTION: Execute opcode with two arguments, one target, and a return
275  *              value.
276  *
277  *****************************************************************************/

279 ACPI_STATUS
280 AcpiExOpcode_2A_1T_1R (
281     ACPI_WALK_STATE         *WalkState)
282 {
283     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
284     ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
285     UINT64                  Index;
286     ACPI_STATUS             Status = AE_OK;
287     ACPI_SIZE               Length = 0;
289     ACPI_SIZE               Length;


290     ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_1T_1R,
291         AcpiPsGetOpcodeName (WalkState->Opcode));


294     /* Execute the opcode */

296     if (WalkState->OpInfo->Flags & AML_MATH)
297     {
298         /* All simple math opcodes (add, etc.) */

300         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
301         if (!ReturnDesc)
302         {
303             Status = AE_NO_MEMORY;
304             goto Cleanup;
305         }

307         ReturnDesc->Integer.Value = AcpiExDoMathOp (WalkState->Opcode,
308                                                     Operand[0]->Integer.Value,
309                                                     Operand[1]->Integer.Value);
310         goto StoreResultToTarget;
311     }

313     switch (WalkState->Opcode)
314     {
315     case AML_MOD_OP: /* Mod (Dividend, Divisor, RemainderResult (ACPI 2.0) */

317         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
318         if (!ReturnDesc)
319         {
320             Status = AE_NO_MEMORY;
```



```
321             goto Cleanup;
322         }

324         /* ReturnDesc will contain the remainder */

326         Status = AcpiUtDivide (Operand[0]->Integer.Value,
327                                Operand[1]->Integer.Value,
328                                NULL,
329                                &ReturnDesc->Integer.Value);
330         break;


332     case AML_CONCAT_OP: /* Concatenate (Data1, Data2, Result) */

334         Status = AcpiExDoConcatenate (Operand[0], Operand[1],
335                     &ReturnDesc, WalkState);
336         break;


338     case AML_TO_STRING_OP: /* ToString (Buffer, Length, Result) (ACPI 2.0) */

339         /*
340          * Input object is guaranteed to be a buffer at this point (it may have
341          * been converted.)  Copy the raw buffer data to a new object of
342          * type String.
343          */

345         /*
346          * Get the length of the new string. It is the smallest of:
347          * 1) Length of the input buffer
348          * 2) Max length as specified in the ToString operator
349          * 3) Length of input buffer up to a zero byte (null terminator)
350          *
351          * NOTE: A length of zero is ok, and will create a zero-length, null
352          *       terminated string.
353          */
359         Length = 0;
354         while ((Length < Operand[0]->Buffer.Length) &&
355                (Length < Operand[1]->Integer.Value) &&
356                (Operand[0]->Buffer.Pointer[Length]))
357         {
358             Length++;
359         }

361         /* Allocate a new string object */

363         ReturnDesc = AcpiUtCreateStringObject (Length);
364         if (!ReturnDesc)
365         {
366             Status = AE_NO_MEMORY;
367             goto Cleanup;
368         }

370         /*
371          * Copy the raw buffer data with no transform.
372          * (NULL terminated already)
373          */
374         ACPI_MEMCPY (ReturnDesc->String.Pointer,
375             Operand[0]->Buffer.Pointer, Length);
376         break;


378     case AML_CONCAT_RES_OP:

380         /* ConcatenateResTemplate (Buffer, Buffer, Result) (ACPI 2.0) */
```

```
382         Status = AcpiExConcatTemplate (Operand[0], Operand[1],
383                     &ReturnDesc, WalkState);
384         break;


386     case AML_INDEX_OP:              /* Index (Source Index Result) */

388         /* Create the internal return object */

390         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_LOCAL_REFERENCE);
391         if (!ReturnDesc)
392         {
393             Status = AE_NO_MEMORY;
394             goto Cleanup;
395         }

397         /* Initialize the Index reference object */

399         Index = Operand[1]->Integer.Value;
400         ReturnDesc->Reference.Value = (UINT32) Index;
401         ReturnDesc->Reference.Class = ACPI_REFCLASS_INDEX;

403         /*
404          * At this point, the Source operand is a String, Buffer, or Package.
405          * Verify that the index is within range.
406          */
407         switch ((Operand[0])->Common.Type)
408         {
409         case ACPI_TYPE_STRING:

411             if (Index >= Operand[0]->String.Length)
412             {
413                 Length = Operand[0]->String.Length;
414                 Status = AE_AML_STRING_LIMIT;
415             }

417             ReturnDesc->Reference.TargetType = ACPI_TYPE_BUFFER_FIELD;
418             break;

420         case ACPI_TYPE_BUFFER:

422             if (Index >= Operand[0]->Buffer.Length)
423             {
424                 Length = Operand[0]->Buffer.Length;
425                 Status = AE_AML_BUFFER_LIMIT;
426             }

428             ReturnDesc->Reference.TargetType = ACPI_TYPE_BUFFER_FIELD;
429             break;

431         case ACPI_TYPE_PACKAGE:

433             if (Index >= Operand[0]->Package.Count)
434             {
435                 Length = Operand[0]->Package.Count;
436                 Status = AE_AML_PACKAGE_LIMIT;
437             }

439             ReturnDesc->Reference.TargetType = ACPI_TYPE_PACKAGE;
440             ReturnDesc->Reference.Where = &Operand[0]->Package.Elements [Index];
441             break;

443         default:

445             Status = AE_AML_INTERNAL;
446             goto Cleanup;
```



```
447         }

449         /* Failure means that the Index was beyond the end of the object */

451         if (ACPI_FAILURE (Status))
452         {
453             ACPI_EXCEPTION ((AE_INFO, Status,
454                 "Index (0x%X%8.8X) is beyond end of object (length 0x%X)",
455                 ACPI_FORMAT_UINT64 (Index), (UINT32) Length));
459                 "Index (0x%8.8X%8.8X) is beyond end of object",
460                 ACPI_FORMAT_UINT64 (Index)));
456             goto Cleanup;
457         }

459         /*
460          * Save the target object and add a reference to it for the life
461          * of the index
462          */
463         ReturnDesc->Reference.Object = Operand[0];
464         AcpiUtAddReference (Operand[0]);

466         /* Store the reference to the Target */

468         Status = AcpiExStore (ReturnDesc, Operand[2], WalkState);

470         /* Return the reference */

472         WalkState->ResultObj = ReturnDesc;
473         goto Cleanup;


475     default:

477         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
478             WalkState->Opcode));
479         Status = AE_AML_BAD_OPCODE;
480         break;
481     }


484 StoreResultToTarget:

486     if (ACPI_SUCCESS (Status))
487     {
488         /*
489          * Store the result of the operation (which is now in ReturnDesc) into
490          * the Target descriptor.
491          */
492         Status = AcpiExStore (ReturnDesc, Operand[2], WalkState);
493         if (ACPI_FAILURE (Status))
494         {
495             goto Cleanup;
496         }

498         if (!WalkState->ResultObj)
499         {
500             WalkState->ResultObj = ReturnDesc;
501         }
502     }


505 Cleanup:

507     /* Delete return object on error */

509     if (ACPI_FAILURE (Status))
```

```
    {
        AcpiUtRemoveReference (ReturnDesc);
        WalkState->ResultObj = NULL;
    }

    return_ACPI_STATUS (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiExOpcode_2A_0T_1R
 *
 * PARAMETERS:  WalkState           - Current walk state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute opcode with 2 arguments, no target, and a return value
 *
 ****************************************************************************/

ACPI_STATUS
AcpiExOpcode_2A_0T_1R (
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
    ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
    ACPI_STATUS             Status = AE_OK;
    BOOLEAN                 LogicalResult = FALSE;


    ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_0T_1R,
        AcpiPsGetOpcodeName (WalkState->Opcode));


    /* Create the internal return object */

    ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
    if (!ReturnDesc)
    {
        Status = AE_NO_MEMORY;
        goto Cleanup;
    }

    /* Execute the Opcode */

    if (WalkState->OpInfo->Flags & AML_LOGICAL_NUMERIC)
    {
        /* LogicalOp  (Operand0, Operand1) */

        Status = AcpiExDoLogicalNumericOp (WalkState->Opcode,
                        Operand[0]->Integer.Value, Operand[1]->Integer.Value,
                        &LogicalResult);
        goto StoreLogicalResult;
    }
    else if (WalkState->OpInfo->Flags & AML_LOGICAL)
    {
        /* LogicalOp  (Operand0, Operand1) */

        Status = AcpiExDoLogicalOp (WalkState->Opcode, Operand[0],
                    Operand[1], &LogicalResult);
        goto StoreLogicalResult;
    }

    switch (WalkState->Opcode)
    {
```



```
    case AML_ACQUIRE_OP:            /* Acquire (MutexObject, Timeout) */

        Status = AcpiExAcquireMutex (Operand[1], Operand[0], WalkState);
        if (Status == AE_TIME)
        {
            LogicalResult = TRUE;       /* TRUE = Acquire timed out */
            Status = AE_OK;
        }
        break;


    case AML_WAIT_OP:               /* Wait (EventObject, Timeout) */

        Status = AcpiExSystemWaitEvent (Operand[1], Operand[0]);
        if (Status == AE_TIME)
        {
            LogicalResult = TRUE;       /* TRUE, Wait timed out */
            Status = AE_OK;
        }
        break;


    default:

        ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
            WalkState->Opcode));
        Status = AE_AML_BAD_OPCODE;
        goto Cleanup;
    }


StoreLogicalResult:
    /*
     * Set return value to according to LogicalResult. logical TRUE (all ones)
     * Default is FALSE (zero)
     */
    if (LogicalResult)
    {
        ReturnDesc->Integer.Value = ACPI_UINT64_MAX;
    }

Cleanup:

    /* Delete return object on error */

    if (ACPI_FAILURE (Status))
    {
        AcpiUtRemoveReference (ReturnDesc);
    }

    /* Save return object on success */

    else
    {
        WalkState->ResultObj = ReturnDesc;
    }

    return_ACPI_STATUS (Status);
}
```

****
    9032 Thu Dec 26 13:49:06 2013
new/usr/src/common/acpica/components/executer/exoparg3.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exoparg3 - AML execution - opcodes with 3 arguments
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXOPARG3_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acinterp.h"
  49 #include "acparser.h"
  50 #include "amlcode.h"


  53 #define _COMPONENT          ACPI_EXECUTER
  54         ACPI_MODULE_NAME    ("exoparg3")


  57 /*!
  58  * Naming convention for AML interpreter execution routines.
```



```
  59  *
  60  * The routines that begin execution of AML opcodes are named with a common
  61  * convention based upon the number of arguments, the number of target operands,
  62  * and whether or not a value is returned:
  63  *
  64  *      AcpiExOpcode_xA_yT_zR
  65  *
  66  * Where:
  67  *
  68  * xA - ARGUMENTS:    The number of arguments (input operands) that are
  69  *                    required for this opcode type (1 through 6 args).
  70  * yT - TARGETS:      The number of targets (output operands) that are required
  71  *                    for this opcode type (0, 1, or 2 targets).
  72  * zR - RETURN VALUE: Indicates whether this opcode type returns a value
  73  *                    as the function return (0 or 1).
  74  *
  75  * The AcpiExOpcode* functions are called via the Dispatcher component with
  76  * fully resolved operands.
  77 !*/


  80 /******************************************************************************
  81  *
  82  * FUNCTION:    AcpiExOpcode_3A_0T_0R
  83  *
  84  * PARAMETERS:  WalkState           - Current walk state
  85  *
  86  * RETURN:      Status
  87  *
  88  * DESCRIPTION: Execute Triadic operator (3 operands)
  89  *
  90  *****************************************************************************/

  92 ACPI_STATUS
  93 AcpiExOpcode_3A_0T_0R (
  94     ACPI_WALK_STATE         *WalkState)
  95 {
  96     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
  97     ACPI_SIGNAL_FATAL_INFO  *Fatal;
  98     ACPI_STATUS             Status = AE_OK;


 101     ACPI_FUNCTION_TRACE_STR (ExOpcode_3A_0T_0R,
 102         AcpiPsGetOpcodeName (WalkState->Opcode));


 105     switch (WalkState->Opcode)
 106     {
 107     case AML_FATAL_OP:          /* Fatal (FatalType  FatalCode  FatalArg) */

 109         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
 110             "FatalOp: Type %X Code %X Arg %X <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n",
 111             (UINT32) Operand[0]->Integer.Value,
 112             (UINT32) Operand[1]->Integer.Value,
 113             (UINT32) Operand[2]->Integer.Value));

 115         Fatal = ACPI_ALLOCATE (sizeof (ACPI_SIGNAL_FATAL_INFO));
 116         if (Fatal)
 117         {
 118             Fatal->Type     = (UINT32) Operand[0]->Integer.Value;
 119             Fatal->Code     = (UINT32) Operand[1]->Integer.Value;
 120             Fatal->Argument = (UINT32) Operand[2]->Integer.Value;
 121         }

 123         /* Always signal the OS! */
```

```
125         Status = AcpiOsSignal (ACPI_SIGNAL_FATAL, Fatal);

127         /* Might return while OS is shutting down, just continue */

129         ACPI_FREE (Fatal);
130         break;


132     default:

134         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
135             WalkState->Opcode));
136         Status = AE_AML_BAD_OPCODE;
137         goto Cleanup;
138     }


141 Cleanup:

143     return_ACPI_STATUS (Status);
144 }


147 /*******************************************************************************
148  *
149  * FUNCTION:    AcpiExOpcode_3A_1T_1R
150  *
151  * PARAMETERS:  WalkState           - Current walk state
152  *
153  * RETURN:      Status
154  *
155  * DESCRIPTION: Execute Triadic operator (3 operands)
156  *
157  ******************************************************************************/

159 ACPI_STATUS
160 AcpiExOpcode_3A_1T_1R (
161     ACPI_WALK_STATE         *WalkState)
162 {
163     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
164     ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
165     char                    *Buffer = NULL;
166     ACPI_STATUS             Status = AE_OK;
167     UINT64                  Index;
168     ACPI_SIZE               Length;


171     ACPI_FUNCTION_TRACE_STR (ExOpcode_3A_1T_1R,
172         AcpiPsGetOpcodeName (WalkState->Opcode));


175     switch (WalkState->Opcode)
176     {
177     case AML_MID_OP:    /* Mid (Source[0], Index[1], Length[2], Result[3]) */

178         /*
179          * Create the return object. The Source operand is guaranteed to be
180          * either a String or a Buffer, so just use its type.
181          */
182         ReturnDesc = AcpiUtCreateInternalObject (
183                         (Operand[0])->Common.Type);
184         if (!ReturnDesc)
185         {
186             Status = AE_NO_MEMORY;
187             goto Cleanup;
188         }
```



```
190         /* Get the Integer values from the objects */

192         Index = Operand[1]->Integer.Value;
193         Length = (ACPI_SIZE) Operand[2]->Integer.Value;

195         /*
196          * If the index is beyond the length of the String/Buffer, or if the
197          * requested length is zero, return a zero-length String/Buffer
198          */
199         if (Index >= Operand[0]->String.Length)
200         {
201             Length = 0;
202         }

204         /* Truncate request if larger than the actual String/Buffer */

206         else if ((Index + Length) > Operand[0]->String.Length)
207         {
208             Length = (ACPI_SIZE) Operand[0]->String.Length -
209                         (ACPI_SIZE) Index;
210         }

212         /* Strings always have a sub-pointer, not so for buffers */

214         switch ((Operand[0])->Common.Type)
215         {
216         case ACPI_TYPE_STRING:

218             /* Always allocate a new buffer for the String */

220             Buffer = ACPI_ALLOCATE_ZEROED ((ACPI_SIZE) Length + 1);
221             if (!Buffer)
222             {
223                 Status = AE_NO_MEMORY;
224                 goto Cleanup;
225             }
226             break;

228         case ACPI_TYPE_BUFFER:

230             /* If the requested length is zero, don't allocate a buffer */

232             if (Length > 0)
233             {
234                 /* Allocate a new buffer for the Buffer */

236                 Buffer = ACPI_ALLOCATE_ZEROED (Length);
237                 if (!Buffer)
238                 {
239                     Status = AE_NO_MEMORY;
240                     goto Cleanup;
241                 }
242             }
243             break;

245         default:                        /* Should not happen */

247             Status = AE_AML_OPERAND_TYPE;
248             goto Cleanup;
249         }

251         if (Buffer)
252         {
253             /* We have a buffer, copy the portion requested */
```

```
255             ACPI_MEMCPY (Buffer, Operand[0]->String.Pointer + Index,
256                         Length);
257         }

259         /* Set the length of the new String/Buffer */

261         ReturnDesc->String.Pointer = Buffer;
262         ReturnDesc->String.Length = (UINT32) Length;

264         /* Mark buffer initialized */

266         ReturnDesc->Buffer.Flags |= AOPOBJ_DATA_VALID;
267         break;


269     default:

271         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
272             WalkState->Opcode));
273         Status = AE_AML_BAD_OPCODE;
274         goto Cleanup;
275     }

277     /* Store the result in the target */

279     Status = AcpiExStore (ReturnDesc, Operand[3], WalkState);

281 Cleanup:

283     /* Delete return object on error */

285     if (ACPI_FAILURE (Status) || WalkState->ResultObj)
286     {
287         AcpiUtRemoveReference (ReturnDesc);
288         WalkState->ResultObj = NULL;
289     }

291     /* Set the return object and exit */

293     else
294     {
295         WalkState->ResultObj = ReturnDesc;
296     }
297     return_ACPI_STATUS (Status);
298 }
```

```
**********************************************************
   11135 Thu Dec 26 13:49:07 2013
new/usr/src/common/acpica/components/executer/exoparg6.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exoparg6 - AML execution - opcodes with 6 arguments
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXOPARG6_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acinterp.h"
  49 #include "acparser.h"
  50 #include "amlcode.h"


  53 #define _COMPONENT          ACPI_EXECUTER
  54         ACPI_MODULE_NAME    ("exoparg6")


  57 /*!
  58  * Naming convention for AML interpreter execution routines.
```



```
  59  *
  60  * The routines that begin execution of AML opcodes are named with a common
  61  * convention based upon the number of arguments, the number of target operands,
  62  * and whether or not a value is returned:
  63  *
  64  *      AcpiExOpcode_xA_yT_zR
  65  *
  66  * Where:
  67  *
  68  * xA - ARGUMENTS:    The number of arguments (input operands) that are
  69  *                    required for this opcode type (1 through 6 args).
  70  * yT - TARGETS:      The number of targets (output operands) that are required
  71  *                    for this opcode type (0, 1, or 2 targets).
  72  * zR - RETURN VALUE: Indicates whether this opcode type returns a value
  73  *                    as the function return (0 or 1).
  74  *
  75  * The AcpiExOpcode* functions are called via the Dispatcher component with
  76  * fully resolved operands.
  77 !*/

  79 /* Local prototypes */

  81 static BOOLEAN
  82 AcpiExDoMatch (
  83     UINT32                  MatchOp,
  84     ACPI_OPERAND_OBJECT     *PackageObj,
  85     ACPI_OPERAND_OBJECT     *MatchObj);


  88 /*******************************************************************************
  89  *
  90  * FUNCTION:    AcpiExDoMatch
  91  *
  92  * PARAMETERS:  MatchOp         - The AML match operand
  93  *              PackageObj      - Object from the target package
  94  *              MatchObj        - Object to be matched
  95  *
  96  * RETURN:      TRUE if the match is successful, FALSE otherwise
  97  *
  98  * DESCRIPTION: Implements the low-level match for the ASL Match operator.
  99  *              Package elements will be implicitly converted to the type of
 100  *              the match object (Integer/Buffer/String).
 101  *
 102  ******************************************************************************/

 104 static BOOLEAN
 105 AcpiExDoMatch (
 106     UINT32                  MatchOp,
 107     ACPI_OPERAND_OBJECT     *PackageObj,
 108     ACPI_OPERAND_OBJECT     *MatchObj)
 109 {
 110     BOOLEAN                 LogicalResult = TRUE;
 111     ACPI_STATUS             Status;


 114     /*
 115      * Note: Since the PackageObj/MatchObj ordering is opposite to that of
 116      * the standard logical operators, we have to reverse them when we call
 117      * DoLogicalOp in order to make the implicit conversion rules work
 118      * correctly. However, this means we have to flip the entire equation
 119      * also. A bit ugly perhaps, but overall, better than fussing the
 120      * parameters around at runtime, over and over again.
 121      *
 122      * Below, P[i] refers to the package element, M refers to the Match object.
 123      */
 124     switch (MatchOp)
```

```
125     {
126     case MATCH_MTR:

128         /* Always true */

130         break;

132     case MATCH_MEQ:

133         /*
134          * True if equal: (P[i] == M)
135          * Change to:     (M == P[i])
136          */
137         Status = AcpiExDoLogicalOp (AML_LEQUAL_OP, MatchObj, PackageObj,
138                     &LogicalResult);
139         if (ACPI_FAILURE (Status))
140         {
141             return (FALSE);
142         }
143         break;

145     case MATCH_MLE:

146         /*
147          * True if less than or equal: (P[i] <= M) (P[i] NotGreater than M)
148          * Change to:                  (M >= P[i]) (M NotLess than P[i])
149          */
150         Status = AcpiExDoLogicalOp (AML_LLESS_OP, MatchObj, PackageObj,
151                     &LogicalResult);
152         if (ACPI_FAILURE (Status))
153         {
154             return (FALSE);
155         }
156         LogicalResult = (BOOLEAN) !LogicalResult;
157         break;

159     case MATCH_MLT:

160         /*
161          * True if less than: (P[i] < M)
162          * Change to:         (M > P[i])
163          */
164         Status = AcpiExDoLogicalOp (AML_LGREATER_OP, MatchObj, PackageObj,
165                     &LogicalResult);
166         if (ACPI_FAILURE (Status))
167         {
168             return (FALSE);
169         }
170         break;

172     case MATCH_MGE:

173         /*
174          * True if greater than or equal: (P[i] >= M) (P[i] NotLess than M)
175          * Change to:                     (M <= P[i]) (M NotGreater than P[i])
176          */
177         Status = AcpiExDoLogicalOp (AML_LGREATER_OP, MatchObj, PackageObj,
178                     &LogicalResult);
179         if (ACPI_FAILURE (Status))
180         {
181             return (FALSE);
182         }
183         LogicalResult = (BOOLEAN)!LogicalResult;
184         break;

186     case MATCH_MGT:
```



```
187         /*
188          * True if greater than: (P[i] > M)
189          * Change to:            (M < P[i])
190          */
191         Status = AcpiExDoLogicalOp (AML_LLESS_OP, MatchObj, PackageObj,
192                     &LogicalResult);
193         if (ACPI_FAILURE (Status))
194         {
195             return (FALSE);
196         }
197         break;

199     default:

201         /* Undefined */

203         return (FALSE);
204     }

206     return (LogicalResult);
212     return LogicalResult;
207 }


210 /*******************************************************************************
211  *
212  * FUNCTION:    AcpiExOpcode_6A_0T_1R
213  *
214  * PARAMETERS:  WalkState           - Current walk state
215  *
216  * RETURN:      Status
217  *
218  * DESCRIPTION: Execute opcode with 6 arguments, no target, and a return value
219  *
220  ******************************************************************************/

222 ACPI_STATUS
223 AcpiExOpcode_6A_0T_1R (
224     ACPI_WALK_STATE         *WalkState)
225 {
226     ACPI_OPERAND_OBJECT     **Operand = &WalkState->Operands[0];
227     ACPI_OPERAND_OBJECT     *ReturnDesc = NULL;
228     ACPI_STATUS             Status = AE_OK;
229     UINT64                  Index;
230     ACPI_OPERAND_OBJECT     *ThisElement;


233     ACPI_FUNCTION_TRACE_STR (ExOpcode_6A_0T_1R,
234         AcpiPsGetOpcodeName (WalkState->Opcode));


237     switch (WalkState->Opcode)
238     {
239     case AML_MATCH_OP:
240         /*
241          * Match (SearchPkg[0], MatchOp1[1], MatchObj1[2],
242          *                      MatchOp2[3], MatchObj2[4], StartIndex[5])
243          */

245         /* Validate both Match Term Operators (MTR, MEQ, etc.) */

247         if ((Operand[1]->Integer.Value > MAX_MATCH_OPERATOR) ||
248             (Operand[3]->Integer.Value > MAX_MATCH_OPERATOR))
249         {
250             ACPI_ERROR ((AE_INFO, "Match operator out of range"));
```

```
251             Status = AE_AML_OPERAND_VALUE;
252             goto Cleanup;
253         }

255         /* Get the package StartIndex, validate against the package length */

257         Index = Operand[5]->Integer.Value;
258         if (Index >= Operand[0]->Package.Count)
259         {
260             ACPI_ERROR ((AE_INFO,
261                 "Index (0x%8.8X%8.8X) beyond package end (0x%X)",
262                 ACPI_FORMAT_UINT64 (Index), Operand[0]->Package.Count));
263             Status = AE_AML_PACKAGE_LIMIT;
264             goto Cleanup;
265         }

267         /* Create an integer for the return value */
268         /* Default return value is ACPI_UINT64_MAX if no match found */

270         ReturnDesc = AcpiUtCreateIntegerObject (ACPI_UINT64_MAX);
271         if (!ReturnDesc)
272         {
273             Status = AE_NO_MEMORY;
274             goto Cleanup;

276         }

278         /*
279          * Examine each element until a match is found. Both match conditions
280          * must be satisfied for a match to occur. Within the loop,
281          * "continue" signifies that the current element does not match
282          * and the next should be examined.
283          *
284          * Upon finding a match, the loop will terminate via "break" at
285          * the bottom. If it terminates "normally", MatchValue will be
286          * ACPI_UINT64_MAX (Ones) (its initial value) indicating that no
287          * match was found.
288          */
289         for ( ; Index < Operand[0]->Package.Count; Index++)
290         {
291             /* Get the current package element */

293             ThisElement = Operand[0]->Package.Elements[Index];

295             /* Treat any uninitialized (NULL) elements as non-matching */

297             if (!ThisElement)
298             {
299                 continue;
300             }

302             /*
303              * Both match conditions must be satisfied. Execution of a continue
304              * (proceed to next iteration of enclosing for loop) signifies a
305              * non-match.
306              */
307             if (!AcpiExDoMatch ((UINT32) Operand[1]->Integer.Value,
308                                 ThisElement, Operand[2]))
309             {
310                 continue;
311             }

313             if (!AcpiExDoMatch ((UINT32) Operand[3]->Integer.Value,
314                                 ThisElement, Operand[4]))
315             {
316                 continue;
```



```
317             }

319             /* Match found: Index is the return value */

321             ReturnDesc->Integer.Value = Index;
322             break;
323         }
324         break;


326     case AML_LOAD_TABLE_OP:

328         Status = AcpiExLoadTableOp (WalkState, &ReturnDesc);
329         break;


331     default:

333         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
334             WalkState->Opcode));
335         Status = AE_AML_BAD_OPCODE;
336         goto Cleanup;
337     }


340 Cleanup:

342     /* Delete return object on error */

344     if (ACPI_FAILURE (Status))
345     {
346         AcpiUtRemoveReference (ReturnDesc);
347     }

349     /* Save return object on success */

351     else
352     {
353         WalkState->ResultObj = ReturnDesc;
354     }

356     return_ACPI_STATUS (Status);
357 }
```

_____***unchanged_portion_omitted_***

****
22243 Thu Dec 26 13:49:07 2013
new/usr/src/common/acpica/components/executer/exprep.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exprep - ACPI AML (p-code) execution - field prep utilities
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXPREP_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acinterp.h"
  49 #include "amlcode.h"
  50 #include "acnamesp.h"
  51 #include "acdispat.h"


  54 #define _COMPONENT          ACPI_EXECUTER
  55         ACPI_MODULE_NAME    ("exprep")

  57 /* Local prototypes */
```



```
  59 static UINT32
  60 AcpiExDecodeFieldAccess (
  61     ACPI_OPERAND_OBJECT     *ObjDesc,
  62     UINT8                   FieldFlags,
  63     UINT32                  *ReturnByteAlignment);


  66 #ifdef ACPI_UNDER_DEVELOPMENT

  68 static UINT32
  69 AcpiExGenerateAccess (
  70     UINT32                  FieldBitOffset,
  71     UINT32                  FieldBitLength,
  72     UINT32                  RegionLength);

  74 /******************************************************************************
  75  *
  76  * FUNCTION:    AcpiExGenerateAccess
  77  *
  78  * PARAMETERS:  FieldBitOffset      - Start of field within parent region/buffer
  79  *              FieldBitLength      - Length of field in bits
  80  *              RegionLength        - Length of parent in bytes
  81  *
  82  * RETURN:      Field granularity (8, 16, 32 or 64) and
  83  *              ByteAlignment (1, 2, 3, or 4)
  84  *
  85  * DESCRIPTION: Generate an optimal access width for fields defined with the
  86  *              AnyAcc keyword.
  87  *
  88  * NOTE: Need to have the RegionLength in order to check for boundary
  89  *       conditions (end-of-region). However, the RegionLength is a deferred
  90  *       operation. Therefore, to complete this implementation, the generation
  91  *       of this access width must be deferred until the region length has
  92  *       been evaluated.
  93  *
  94  *****************************************************************************/

  96 static UINT32
  97 AcpiExGenerateAccess (
  98     UINT32                  FieldBitOffset,
  99     UINT32                  FieldBitLength,
 100     UINT32                  RegionLength)
 101 {
 102     UINT32                  FieldByteLength;
 103     UINT32                  FieldByteOffset;
 104     UINT32                  FieldByteEndOffset;
 105     UINT32                  AccessByteWidth;
 106     UINT32                  FieldStartOffset;
 107     UINT32                  FieldEndOffset;
 108     UINT32                  MinimumAccessWidth = 0xFFFFFFFF;
 109     UINT32                  MinimumAccesses = 0xFFFFFFFF;
 110     UINT32                  Accesses;


 113     ACPI_FUNCTION_TRACE (ExGenerateAccess);


 116     /* Round Field start offset and length to "minimal" byte boundaries */

 118     FieldByteOffset    = ACPI_DIV_8 (ACPI_ROUND_DOWN (FieldBitOffset, 8));
 119     FieldByteEndOffset = ACPI_DIV_8 (ACPI_ROUND_UP   (FieldBitLength +
 120                                                        FieldBitOffset, 8));
 121     FieldByteLength    = FieldByteEndOffset - FieldByteOffset;

 123     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
 124         "Bit length %u, Bit offset %u\n",
```

```
125         FieldBitLength, FieldBitOffset));

127     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
128         "Byte Length %u, Byte Offset %u, End Offset %u\n",
129         FieldByteLength, FieldByteOffset, FieldByteEndOffset));

131     /*
132      * Iterative search for the maximum access width that is both aligned
133      * and does not go beyond the end of the region
134      *
135      * Start at ByteAcc and work upwards to QwordAcc max. (1,2,4,8 bytes)
136      */
137     for (AccessByteWidth = 1; AccessByteWidth <= 8; AccessByteWidth <<= 1)
138     {
139         /*
140          * 1) Round end offset up to next access boundary and make sure that
141          *    this does not go beyond the end of the parent region.
142          * 2) When the Access width is greater than the FieldByteLength, we
143          *    are done. (This does not optimize for the perfectly aligned
144          *    case yet).
145          */
146         if (ACPI_ROUND_UP (FieldByteEndOffset, AccessByteWidth) <= RegionLength)
147         {
148             FieldStartOffset =
149                 ACPI_ROUND_DOWN (FieldByteOffset, AccessByteWidth) /
150                 AccessByteWidth;

152             FieldEndOffset =
153                 ACPI_ROUND_UP ((FieldByteLength + FieldByteOffset),
154                     AccessByteWidth) / AccessByteWidth;

156             Accesses = FieldEndOffset - FieldStartOffset;

158             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
159                 "AccessWidth %u end is within region\n", AccessByteWidth));

161             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
162                 "Field Start %u, Field End %u -- requires %u accesses\n",
163                 FieldStartOffset, FieldEndOffset, Accesses));

165             /* Single access is optimal */

167             if (Accesses <= 1)
168             {
169                 ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
170                     "Entire field can be accessed with one operation of size %u\
171                     AccessByteWidth));
172                 return_VALUE (AccessByteWidth);
173             }

175             /*
176              * Fits in the region, but requires more than one read/write.
177              * try the next wider access on next iteration
178              */
179             if (Accesses < MinimumAccesses)
180             {
181                 MinimumAccesses    = Accesses;
182                 MinimumAccessWidth = AccessByteWidth;
183             }
184         }
185         else
186         {
187             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
188                 "AccessWidth %u end is NOT within region\n", AccessByteWidth));
189             if (AccessByteWidth == 1)
190             {
```



```
191                 ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
192                     "Field goes beyond end-of-region!\n"));

194                 /* Field does not fit in the region at all */

196                 return_VALUE (0);
197             }

199             /*
200              * This width goes beyond the end-of-region, back off to
201              * previous access
202              */
203             ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
204                 "Backing off to previous optimal access width of %u\n",
205                 MinimumAccessWidth));
206             return_VALUE (MinimumAccessWidth);
207         }
208     }

210     /*
211      * Could not read/write field with one operation,
212      * just use max access width
213      */
214     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
215         "Cannot access field in one operation, using width 8\n"));
216     return_VALUE (8);
217 }
218 #endif /* ACPI_UNDER_DEVELOPMENT */


221 /*******************************************************************************
222  *
223  * FUNCTION:    AcpiExDecodeFieldAccess
224  *
225  * PARAMETERS:  ObjDesc             - Field object
226  *              FieldFlags          - Encoded fieldflags (contains access bits)
227  *              ReturnByteAlignment - Where the byte alignment is returned
228  *
229  * RETURN:      Field granularity (8, 16, 32 or 64) and
230  *              ByteAlignment (1, 2, 3, or 4)
231  *
232  * DESCRIPTION: Decode the AccessType bits of a field definition.
233  *
234  ******************************************************************************/

236 static UINT32
237 AcpiExDecodeFieldAccess (
238     ACPI_OPERAND_OBJECT     *ObjDesc,
239     UINT8                   FieldFlags,
240     UINT32                  *ReturnByteAlignment)
241 {
242     UINT32                  Access;
243     UINT32                  ByteAlignment;
244     UINT32                  BitLength;


247     ACPI_FUNCTION_TRACE (ExDecodeFieldAccess);


250     Access = (FieldFlags & AML_FIELD_ACCESS_TYPE_MASK);

252     switch (Access)
253     {
254     case AML_FIELD_ACCESS_ANY:

256 #ifdef ACPI_UNDER_DEVELOPMENT
```

```
257         ByteAlignment =
258             AcpiExGenerateAccess (ObjDesc->CommonField.StartFieldBitOffset,
259                 ObjDesc->CommonField.BitLength,
260                 0xFFFFFFFF /* Temp until we pass RegionLength as parameter */);
261         BitLength = ByteAlignment * 8;
262 #endif

264         ByteAlignment = 1;
265         BitLength = 8;
266         break;

268     case AML_FIELD_ACCESS_BYTE:
269     case AML_FIELD_ACCESS_BUFFER:   /* ACPI 2.0 (SMBus Buffer) */

271         ByteAlignment = 1;
272         BitLength     = 8;
273         break;

275     case AML_FIELD_ACCESS_WORD:

277         ByteAlignment = 2;
278         BitLength     = 16;
279         break;

281     case AML_FIELD_ACCESS_DWORD:

283         ByteAlignment = 4;
284         BitLength     = 32;
285         break;

287     case AML_FIELD_ACCESS_QWORD:    /* ACPI 2.0 */

289         ByteAlignment = 8;
290         BitLength     = 64;
291         break;

293     default:

295         /* Invalid field access type */

297         ACPI_ERROR ((AE_INFO,
298             "Unknown field access type 0x%X",
299             Access));
300         return_UINT32 (0);
301     }

303     if (ObjDesc->Common.Type == ACPI_TYPE_BUFFER_FIELD)
304     {
305         /*
306          * BufferField access can be on any byte boundary, so the
307          * ByteAlignment is always 1 byte -- regardless of any ByteAlignment
308          * implied by the field access type.
309          */
310         ByteAlignment = 1;
311     }

313     *ReturnByteAlignment = ByteAlignment;
314     return_UINT32 (BitLength);
315 }
```

unchanged_portion_omitted_

```
417 /*******************************************************************************
418  *
419  * FUNCTION:    AcpiExPrepFieldValue
420  *
```



```
421  * PARAMETERS:  Info    - Contains all field creation info
422  *
423  * RETURN:      Status
424  *
425  * DESCRIPTION: Construct an object of type ACPI_OPERAND_OBJECT with a
426  *              subtype of DefField and connect it to the parent Node.
420  * DESCRIPTION: Construct an ACPI_OPERAND_OBJECT of type DefField and
421  *              connect it to the parent Node.
427  *
428  ******************************************************************************/

430 ACPI_STATUS
431 AcpiExPrepFieldValue (
432     ACPI_CREATE_FIELD_INFO  *Info)
433 {
434     ACPI_OPERAND_OBJECT     *ObjDesc;
435     ACPI_OPERAND_OBJECT     *SecondDesc = NULL;
436     ACPI_STATUS             Status;
437     UINT32                  AccessByteWidth;
438     UINT32                  Type;


441     ACPI_FUNCTION_TRACE (ExPrepFieldValue);


444     /* Parameter validation */

446     if (Info->FieldType != ACPI_TYPE_LOCAL_INDEX_FIELD)
447     {
448         if (!Info->RegionNode)
449         {
450             ACPI_ERROR ((AE_INFO, "Null RegionNode"));
451             return_ACPI_STATUS (AE_AML_NO_OPERAND);
452         }

454         Type = AcpiNsGetType (Info->RegionNode);
455         if (Type != ACPI_TYPE_REGION)
456         {
457             ACPI_ERROR ((AE_INFO, "Needed Region, found type 0x%X (%s)",
458                 Type, AcpiUtGetTypeName (Type)));

460             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
461         }
462     }

464     /* Allocate a new field object */

466     ObjDesc = AcpiUtCreateInternalObject (Info->FieldType);
467     if (!ObjDesc)
468     {
469         return_ACPI_STATUS (AE_NO_MEMORY);
470     }

472     /* Initialize areas of the object that are common to all fields */

474     ObjDesc->CommonField.Node = Info->FieldNode;
475     Status = AcpiExPrepCommonFieldObject (ObjDesc,
476                 Info->FieldFlags, Info->Attribute,
477                 Info->FieldBitPosition, Info->FieldBitLength);
478     if (ACPI_FAILURE (Status))
479     {
480         AcpiUtDeleteObjectDesc (ObjDesc);
481         return_ACPI_STATUS (Status);
482     }

484     /* Initialize areas of the object that are specific to the field type */
```

```
486     switch (Info->FieldType)
487     {
488     case ACPI_TYPE_LOCAL_REGION_FIELD:

490         ObjDesc->Field.RegionObj = AcpiNsGetAttachedObject (Info->RegionNode);

492         /* Fields specific to GenericSerialBus fields */

494         ObjDesc->Field.AccessLength = Info->AccessLength;

496         if (Info->ConnectionNode)
497         {
498             SecondDesc = Info->ConnectionNode->Object;
499             if (!(SecondDesc->Common.Flags & AOPOBJ_DATA_VALID))
500             {
501                 Status = AcpiDsGetBufferArguments (SecondDesc);
502                 if (ACPI_FAILURE (Status))
503                 {
504                     AcpiUtDeleteObjectDesc (ObjDesc);
505                     return_ACPI_STATUS (Status);
506                 }
507             }

509             ObjDesc->Field.ResourceBuffer = SecondDesc->Buffer.Pointer;
510             ObjDesc->Field.ResourceLength = (UINT16) SecondDesc->Buffer.Length;
511         }
512         else if (Info->ResourceBuffer)
513         {
514             ObjDesc->Field.ResourceBuffer = Info->ResourceBuffer;
515             ObjDesc->Field.ResourceLength = Info->ResourceLength;
516         }

518         /* Allow full data read from EC address space */

520         if ((ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_EC) &&
521             (ObjDesc->CommonField.BitLength > 8))
522         {
523             AccessByteWidth = ACPI_ROUND_BITS_UP_TO_BYTES (
524                 ObjDesc->CommonField.BitLength);

526             /* Maximum byte width supported is 255 */

528             if (AccessByteWidth < 256)
529             {
530                 ObjDesc->CommonField.AccessByteWidth = (UINT8) AccessByteWidth;
531             }
532         }

534         /* An additional reference for the container */

536         AcpiUtAddReference (ObjDesc->Field.RegionObj);

538         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
539             "RegionField: BitOff %X, Off %X, Gran %X, Region %p\n",
540             ObjDesc->Field.StartFieldBitOffset, ObjDesc->Field.BaseByteOffset,
541             ObjDesc->Field.AccessByteWidth, ObjDesc->Field.RegionObj));
542         break;


544     case ACPI_TYPE_LOCAL_BANK_FIELD:

546         ObjDesc->BankField.Value = Info->BankValue;
547         ObjDesc->BankField.RegionObj =
548             AcpiNsGetAttachedObject (Info->RegionNode);
549         ObjDesc->BankField.BankObj =
```



```
550             AcpiNsGetAttachedObject (Info->RegisterNode);

552         /* An additional reference for the attached objects */

554         AcpiUtAddReference (ObjDesc->BankField.RegionObj);
555         AcpiUtAddReference (ObjDesc->BankField.BankObj);

557         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
558             "Bank Field: BitOff %X, Off %X, Gran %X, Region %p, BankReg %p\n",
559             ObjDesc->BankField.StartFieldBitOffset,
560             ObjDesc->BankField.BaseByteOffset,
561             ObjDesc->Field.AccessByteWidth,
562             ObjDesc->BankField.RegionObj,
563             ObjDesc->BankField.BankObj));

565         /*
566          * Remember location in AML stream of the field unit
567          * opcode and operands -- since the BankValue
568          * operands must be evaluated.
569          */
570         SecondDesc = ObjDesc->Common.NextObject;
571         SecondDesc->Extra.AmlStart = ACPI_CAST_PTR (ACPI_PARSE_OBJECT,
572             Info->DataRegisterNode)->Named.Data;
573         SecondDesc->Extra.AmlLength = ACPI_CAST_PTR (ACPI_PARSE_OBJECT,
574             Info->DataRegisterNode)->Named.Length;

576         break;


578     case ACPI_TYPE_LOCAL_INDEX_FIELD:

580         /* Get the Index and Data registers */

582         ObjDesc->IndexField.IndexObj =
583             AcpiNsGetAttachedObject (Info->RegisterNode);
584         ObjDesc->IndexField.DataObj =
585             AcpiNsGetAttachedObject (Info->DataRegisterNode);

587         if (!ObjDesc->IndexField.DataObj || !ObjDesc->IndexField.IndexObj)
588         {
589             ACPI_ERROR ((AE_INFO, "Null Index Object during field prep"));
590             AcpiUtDeleteObjectDesc (ObjDesc);
591             return_ACPI_STATUS (AE_AML_INTERNAL);
592         }

594         /* An additional reference for the attached objects */

596         AcpiUtAddReference (ObjDesc->IndexField.DataObj);
597         AcpiUtAddReference (ObjDesc->IndexField.IndexObj);

599         /*
600          * April 2006: Changed to match MS behavior
601          *
602          * The value written to the Index register is the byte offset of the
603          * target field in units of the granularity of the IndexField
604          *
605          * Previously, the value was calculated as an index in terms of the
606          * width of the Data register, as below:
607          *
608          *      ObjDesc->IndexField.Value = (UINT32)
609          *          (Info->FieldBitPosition / ACPI_MUL_8 (
610          *              ObjDesc->Field.AccessByteWidth));
611          *
612          * February 2006: Tried value as a byte offset:
613          *      ObjDesc->IndexField.Value = (UINT32)
614          *          ACPI_DIV_8 (Info->FieldBitPosition);
```

```
615          */
616         ObjDesc->IndexField.Value = (UINT32) ACPI_ROUND_DOWN (
617             ACPI_DIV_8 (Info->FieldBitPosition),
618             ObjDesc->IndexField.AccessByteWidth);
619
620         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
621             "IndexField: BitOff %X, Off %X, Value %X, Gran %X, Index %p, Data %p
622             ObjDesc->IndexField.StartFieldBitOffset,
623             ObjDesc->IndexField.BaseByteOffset,
624             ObjDesc->IndexField.Value,
625             ObjDesc->Field.AccessByteWidth,
626             ObjDesc->IndexField.IndexObj,
627             ObjDesc->IndexField.DataObj));
628         break;
629
630     default:
631
632         /* No other types should get here */
633
634         break;
635     }
636
637     /*
638      * Store the constructed descriptor (ObjDesc) into the parent Node,
639      * preserving the current type of that NamedObj.
640      */
641     Status = AcpiNsAttachObject (Info->FieldNode, ObjDesc,
642                 AcpiNsGetType (Info->FieldNode));
643
644     ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD, "Set NamedObj %p [%4.4s], ObjDesc %p\n",
645         Info->FieldNode, AcpiUtGetNodeName (Info->FieldNode), ObjDesc));
646
647     /* Remove local reference to the object */
648
649     AcpiUtRemoveReference (ObjDesc);
650     return_ACPI_STATUS (Status);
651 }
```

****
17443 Thu Dec 26 13:49:08 2013
new/usr/src/common/acpica/components/executer/exregion.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exregion - ACPI default OpRegion (address space) handlers
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __EXREGION_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acinterp.h"


  52 #define _COMPONENT          ACPI_EXECUTER
  53         ACPI_MODULE_NAME    ("exregion")


  56 /******************************************************************************
  57  *
  58  * FUNCTION:    AcpiExSystemMemorySpaceHandler
```



```
  59  *
  60  * PARAMETERS:  Function            - Read or Write operation
  61  *              Address             - Where in the space to read or write
  62  *              BitWidth            - Field width in bits (8, 16, or 32)
  63  *              Value               - Pointer to in or out value
  64  *              HandlerContext      - Pointer to Handler's context
  65  *              RegionContext       - Pointer to context specific to the
  66  *                                    accessed region
  67  *
  68  * RETURN:      Status
  69  *
  70  * DESCRIPTION: Handler for the System Memory address space (Op Region)
  71  *
  72  *****************************************************************************/

  74 ACPI_STATUS
  75 AcpiExSystemMemorySpaceHandler (
  76     UINT32                  Function,
  77     ACPI_PHYSICAL_ADDRESS   Address,
  78     UINT32                  BitWidth,
  79     UINT64                  *Value,
  80     void                    *HandlerContext,
  81     void                    *RegionContext)
  82 {
  83     ACPI_STATUS             Status = AE_OK;
  84     void                    *LogicalAddrPtr = NULL;
  85     ACPI_MEM_SPACE_CONTEXT  *MemInfo = RegionContext;
  86     UINT32                  Length;
  87     ACPI_SIZE               MapLength;
  88     ACPI_SIZE               PageBoundaryMapLength;
  89 #ifdef ACPI_MISALIGNMENT_NOT_SUPPORTED
  90     UINT32                  Remainder;
  91 #endif


  94     ACPI_FUNCTION_TRACE (ExSystemMemorySpaceHandler);


  97     /* Validate and translate the bit width */

  99     switch (BitWidth)
 100     {
 101     case 8:

 103         Length = 1;
 104         break;

 106     case 16:

 108         Length = 2;
 109         break;

 111     case 32:

 113         Length = 4;
 114         break;

 116     case 64:

 118         Length = 8;
 119         break;

 121     default:

 123         ACPI_ERROR ((AE_INFO, "Invalid SystemMemory width %u",
 124             BitWidth));
```

```
125         return_ACPI_STATUS (AE_AML_OPERAND_VALUE);
126     }

128 #ifdef ACPI_MISALIGNMENT_NOT_SUPPORTED
129     /*
130      * Hardware does not support non-aligned data transfers, we must verify
131      * the request.
132      */
133     (void) AcpiUtShortDivide ((UINT64) Address, Length, NULL, &Remainder);
134     if (Remainder != 0)
135     {
136         return_ACPI_STATUS (AE_AML_ALIGNMENT);
137     }
138 #endif

140     /*
141      * Does the request fit into the cached memory mapping?
142      * Is 1) Address below the current mapping? OR
143      *    2) Address beyond the current mapping?
144      */
145     if ((Address < MemInfo->MappedPhysicalAddress) ||
146         (((UINT64) Address + Length) >
147             ((UINT64)
148             MemInfo->MappedPhysicalAddress + MemInfo->MappedLength)))
149     {
150         /*
151          * The request cannot be resolved by the current memory mapping;
152          * Delete the existing mapping and create a new one.
153          */
154         if (MemInfo->MappedLength)
155         {
156             /* Valid mapping, delete it */

158             AcpiOsUnmapMemory (MemInfo->MappedLogicalAddress,
159                 MemInfo->MappedLength);
160         }

162         /*
163          * October 2009: Attempt to map from the requested address to the
164          * end of the region. However, we will never map more than one
165          * page, nor will we cross a page boundary.
166          */
167         MapLength = (ACPI_SIZE)
168             ((MemInfo->Address + MemInfo->Length) - Address);

170         /*
171          * If mapping the entire remaining portion of the region will cross
172          * a page boundary, just map up to the page boundary, do not cross.
173          * On some systems, crossing a page boundary while mapping regions
174          * can cause warnings if the pages have different attributes
175          * due to resource management.
176          *
177          * This has the added benefit of constraining a single mapping to
178          * one page, which is similar to the original code that used a 4k
179          * maximum window.
180          */
181         PageBoundaryMapLength =
182             ACPI_ROUND_UP (Address, ACPI_DEFAULT_PAGE_SIZE) - Address;
183         if (PageBoundaryMapLength == 0)
184         {
185             PageBoundaryMapLength = ACPI_DEFAULT_PAGE_SIZE;
186         }

188         if (MapLength > PageBoundaryMapLength)
189         {
190             MapLength = PageBoundaryMapLength;
```



```
191         }

193         /* Create a new mapping starting at the address given */

195         MemInfo->MappedLogicalAddress = AcpiOsMapMemory (
196             (ACPI_PHYSICAL_ADDRESS) Address, MapLength);
197         if (!MemInfo->MappedLogicalAddress)
198         {
199             ACPI_ERROR ((AE_INFO,
200                 "Could not map memory at 0x%8.8X%8.8X, size %u",
201                 ACPI_FORMAT_NATIVE_UINT (Address), (UINT32) MapLength));
202             MemInfo->MappedLength = 0;
203             return_ACPI_STATUS (AE_NO_MEMORY);
204         }

206         /* Save the physical address and mapping size */

208         MemInfo->MappedPhysicalAddress = Address;
209         MemInfo->MappedLength = MapLength;
210     }

212     /*
213      * Generate a logical pointer corresponding to the address we want to
214      * access
215      */
216     LogicalAddrPtr = MemInfo->MappedLogicalAddress +
217         ((UINT64) Address - (UINT64) MemInfo->MappedPhysicalAddress);

219     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
220         "System-Memory (width %u) R/W %u Address=%8.8X%8.8X\n",
221         BitWidth, Function, ACPI_FORMAT_NATIVE_UINT (Address)));

223     /*
224      * Perform the memory read or write
225      *
226      * Note: For machines that do not support non-aligned transfers, the target
227      * address was checked for alignment above. We do not attempt to break the
228      * transfer up into smaller (byte-size) chunks because the AML specifically
229      * asked for a transfer width that the hardware may require.
230      */
231     switch (Function)
232     {
233     case ACPI_READ:

235         *Value = 0;
236         switch (BitWidth)
237         {
238         case 8:

240             *Value = (UINT64) ACPI_GET8 (LogicalAddrPtr);
241             break;

243         case 16:

245             *Value = (UINT64) ACPI_GET16 (LogicalAddrPtr);
246             break;

248         case 32:

250             *Value = (UINT64) ACPI_GET32 (LogicalAddrPtr);
251             break;

253         case 64:

255             *Value = (UINT64) ACPI_GET64 (LogicalAddrPtr);
256             break;
```

```
258         default:

260             /* BitWidth was already validated */

262             break;
263         }
264         break;

266     case ACPI_WRITE:

268         switch (BitWidth)
269         {
270         case 8:

272             ACPI_SET8 (LogicalAddrPtr, *Value);
261             ACPI_SET8 (LogicalAddrPtr) = (UINT8) *Value;
273             break;

275         case 16:

277             ACPI_SET16 (LogicalAddrPtr, *Value);
265             ACPI_SET16 (LogicalAddrPtr) = (UINT16) *Value;
278             break;

280         case 32:

282             ACPI_SET32 (LogicalAddrPtr, *Value);
269             ACPI_SET32 ( LogicalAddrPtr) = (UINT32) *Value;
283             break;

285         case 64:

287             ACPI_SET64 (LogicalAddrPtr, *Value);
273             ACPI_SET64 (LogicalAddrPtr) = (UINT64) *Value;
288             break;

290         default:

292             /* BitWidth was already validated */

294             break;
295         }
296         break;

298     default:

300         Status = AE_BAD_PARAMETER;
301         break;
302     }

304     return_ACPI_STATUS (Status);
305 }


308 /*******************************************************************************
309  *
310  * FUNCTION:    AcpiExSystemIoSpaceHandler
311  *
312  * PARAMETERS:  Function            - Read or Write operation
313  *              Address             - Where in the space to read or write
314  *              BitWidth            - Field width in bits (8, 16, or 32)
315  *              Value               - Pointer to in or out value
316  *              HandlerContext      - Pointer to Handler's context
317  *              RegionContext       - Pointer to context specific to the
318  *                                    accessed region
```



```
319  *
320  * RETURN:      Status
321  *
322  * DESCRIPTION: Handler for the System IO address space (Op Region)
323  *
324  ******************************************************************************/

326 ACPI_STATUS
327 AcpiExSystemIoSpaceHandler (
328     UINT32                  Function,
329     ACPI_PHYSICAL_ADDRESS   Address,
330     UINT32                  BitWidth,
331     UINT64                  *Value,
332     void                    *HandlerContext,
333     void                    *RegionContext)
334 {
335     ACPI_STATUS             Status = AE_OK;
336     UINT32                  Value32;


339     ACPI_FUNCTION_TRACE (ExSystemIoSpaceHandler);


342     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
343         "System-IO (width %u) R/W %u Address=%8.8X%8.8X\n",
344         BitWidth, Function, ACPI_FORMAT_NATIVE_UINT (Address)));

346     /* Decode the function parameter */

348     switch (Function)
349     {
350     case ACPI_READ:

352         Status = AcpiHwReadPort ((ACPI_IO_ADDRESS) Address,
353                     &Value32, BitWidth);
354         *Value = Value32;
355         break;

357     case ACPI_WRITE:

359         Status = AcpiHwWritePort ((ACPI_IO_ADDRESS) Address,
360                     (UINT32) *Value, BitWidth);
361         break;

363     default:

365         Status = AE_BAD_PARAMETER;
366         break;
367     }

369     return_ACPI_STATUS (Status);
370 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    9366 Thu Dec 26 13:49:08 2013
new/usr/src/common/acpica/components/executer/exresnte.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exresnte - AML Interpreter object resolution
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXRESNTE_C__

#include "acpi.h"
#include "accommon.h"
#include "acdispat.h"
#include "acinterp.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exresnte")


/*******************************************************************************
```



```
 *
 * FUNCTION:    AcpiExResolveNodeToValue
 *
 * PARAMETERS:  ObjectPtr       - Pointer to a location that contains
 *                                a pointer to a NS node, and will receive a
 *                                pointer to the resolved object.
 *              WalkState       - Current state. Valid only if executing AML
 *                                code. NULL if simply resolving an object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Resolve a Namespace node to a valued object
 *
 * Note: for some of the data types, the pointer attached to the Node
 * can be either a pointer to an actual internal object or a pointer into the
 * AML stream itself. These types are currently:
 *
 *      ACPI_TYPE_INTEGER
 *      ACPI_TYPE_STRING
 *      ACPI_TYPE_BUFFER
 *      ACPI_TYPE_MUTEX
 *      ACPI_TYPE_PACKAGE
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExResolveNodeToValue (
    ACPI_NAMESPACE_NODE     **ObjectPtr,
    ACPI_WALK_STATE         *WalkState)

{
    ACPI_STATUS             Status = AE_OK;
    ACPI_OPERAND_OBJECT     *SourceDesc;
    ACPI_OPERAND_OBJECT     *ObjDesc = NULL;
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_OBJECT_TYPE        EntryType;


    ACPI_FUNCTION_TRACE (ExResolveNodeToValue);


    /*
     * The stack pointer points to a ACPI_NAMESPACE_NODE (Node). Get the
     * object that is attached to the Node.
     */
    Node       = *ObjectPtr;
    SourceDesc = AcpiNsGetAttachedObject (Node);
    EntryType  = AcpiNsGetType ((ACPI_HANDLE) Node);

    ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Entry=%p SourceDesc=%p [%s]\n",
         Node, SourceDesc, AcpiUtGetTypeName (EntryType)));

    if ((EntryType == ACPI_TYPE_LOCAL_ALIAS) ||
        (EntryType == ACPI_TYPE_LOCAL_METHOD_ALIAS))
    {
        /* There is always exactly one level of indirection */

        Node       = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Node->Object);
        SourceDesc = AcpiNsGetAttachedObject (Node);
        EntryType  = AcpiNsGetType ((ACPI_HANDLE) Node);
        *ObjectPtr = Node;
    }

    /*
     * Several object types require no further processing:
     * 1) Device/Thermal objects don't have a "real" subobject, return the Node
```

```
124      * 2) Method locals and arguments have a pseudo-Node
125      * 3) 10/2007: Added method type to assist with Package construction.
126      */
127     if ((EntryType == ACPI_TYPE_DEVICE)  ||
128         (EntryType == ACPI_TYPE_THERMAL) ||
129         (EntryType == ACPI_TYPE_METHOD)  ||
130         (Node->Flags & (ANOBJ_METHOD_ARG | ANOBJ_METHOD_LOCAL)))
131     {
132         return_ACPI_STATUS (AE_OK);
133     }

135     if (!SourceDesc)
136     {
137         ACPI_ERROR ((AE_INFO, "No object attached to node [%4.4s] %p",
138             Node->Name.Ascii, Node));
138         ACPI_ERROR ((AE_INFO, "No object attached to node %p",
139             Node));
139         return_ACPI_STATUS (AE_AML_NO_OPERAND);
140     }

142     /*
143      * Action is based on the type of the Node, which indicates the type
144      * of the attached object or pointer
145      */
146     switch (EntryType)
147     {
148     case ACPI_TYPE_PACKAGE:

150         if (SourceDesc->Common.Type != ACPI_TYPE_PACKAGE)
151         {
152             ACPI_ERROR ((AE_INFO, "Object not a Package, type %s",
153                 AcpiUtGetObjectTypeName (SourceDesc)));
154             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
155         }

157         Status = AcpiDsGetPackageArguments (SourceDesc);
158         if (ACPI_SUCCESS (Status))
159         {
160             /* Return an additional reference to the object */

162             ObjDesc = SourceDesc;
163             AcpiUtAddReference (ObjDesc);
164         }
165         break;


167     case ACPI_TYPE_BUFFER:

169         if (SourceDesc->Common.Type != ACPI_TYPE_BUFFER)
170         {
171             ACPI_ERROR ((AE_INFO, "Object not a Buffer, type %s",
172                 AcpiUtGetObjectTypeName (SourceDesc)));
173             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
174         }

176         Status = AcpiDsGetBufferArguments (SourceDesc);
177         if (ACPI_SUCCESS (Status))
178         {
179             /* Return an additional reference to the object */

181             ObjDesc = SourceDesc;
182             AcpiUtAddReference (ObjDesc);
183         }
184         break;
```



```
186     case ACPI_TYPE_STRING:

188         if (SourceDesc->Common.Type != ACPI_TYPE_STRING)
189         {
190             ACPI_ERROR ((AE_INFO, "Object not a String, type %s",
191                 AcpiUtGetObjectTypeName (SourceDesc)));
192             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
193         }

195         /* Return an additional reference to the object */

197         ObjDesc = SourceDesc;
198         AcpiUtAddReference (ObjDesc);
199         break;


201     case ACPI_TYPE_INTEGER:

203         if (SourceDesc->Common.Type != ACPI_TYPE_INTEGER)
204         {
205             ACPI_ERROR ((AE_INFO, "Object not a Integer, type %s",
206                 AcpiUtGetObjectTypeName (SourceDesc)));
207             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
208         }

210         /* Return an additional reference to the object */

212         ObjDesc = SourceDesc;
213         AcpiUtAddReference (ObjDesc);
214         break;


216     case ACPI_TYPE_BUFFER_FIELD:
217     case ACPI_TYPE_LOCAL_REGION_FIELD:
218     case ACPI_TYPE_LOCAL_BANK_FIELD:
219     case ACPI_TYPE_LOCAL_INDEX_FIELD:

221         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
222             "FieldRead Node=%p SourceDesc=%p Type=%X\n",
223             Node, SourceDesc, EntryType));

225         Status = AcpiExReadDataFromField (WalkState, SourceDesc, &ObjDesc);
226         break;

228     /* For these objects, just return the object attached to the Node */

230     case ACPI_TYPE_MUTEX:
231     case ACPI_TYPE_POWER:
232     case ACPI_TYPE_PROCESSOR:
233     case ACPI_TYPE_EVENT:
234     case ACPI_TYPE_REGION:

236         /* Return an additional reference to the object */

238         ObjDesc = SourceDesc;
239         AcpiUtAddReference (ObjDesc);
240         break;

242     /* TYPE_ANY is untyped, and thus there is no object associated with it */

244     case ACPI_TYPE_ANY:

246         ACPI_ERROR ((AE_INFO,
247             "Untyped entry %p, no attached object!", Node));

249         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);  /* Cannot be AE_TYPE */
```

```
251     case ACPI_TYPE_LOCAL_REFERENCE:

253         switch (SourceDesc->Reference.Class)
254         {
255         case ACPI_REFCLASS_TABLE:   /* This is a DdbHandle */
256         case ACPI_REFCLASS_REFOF:
257         case ACPI_REFCLASS_INDEX:

259             /* Return an additional reference to the object */

261             ObjDesc = SourceDesc;
262             AcpiUtAddReference (ObjDesc);
263             break;

265         default:

267             /* No named references are allowed here */

269             ACPI_ERROR ((AE_INFO,
270                 "Unsupported Reference type 0x%X",
271                 SourceDesc->Reference.Class));

273             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
274         }
275         break;


277     default:

279         /* Default case is for unknown types */

281         ACPI_ERROR ((AE_INFO,
282             "Node %p - Unknown object type 0x%X",
283             Node, EntryType));

285         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);

287     } /* switch (EntryType) */


290     /* Return the object descriptor */

292     *ObjectPtr = (void *) ObjDesc;
293     return_ACPI_STATUS (Status);
294 }
```

****
17048 Thu Dec 26 13:49:09 2013
new/usr/src/common/acpica/components/executer/exresolv.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exresolv - AML Interpreter object resolution
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  44 #define __EXRESOLV_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "amlcode.h"
  49 #include "acdispat.h"
  50 #include "acinterp.h"
  51 #include "acnamesp.h"


  54 #define _COMPONENT          ACPI_EXECUTER
  55         ACPI_MODULE_NAME    ("exresolv")

  57 /* Local prototypes */
```



```
  59 static ACPI_STATUS
  60 AcpiExResolveObjectToValue (
  61     ACPI_OPERAND_OBJECT     **StackPtr,
  62     ACPI_WALK_STATE         *WalkState);


  65 /*******************************************************************************
  66  *
  67  * FUNCTION:    AcpiExResolveToValue
  68  *
  69  * PARAMETERS:  **StackPtr          - Points to entry on ObjStack, which can
  70  *                                    be either an (ACPI_OPERAND_OBJECT *)
  71  *                                    or an ACPI_HANDLE.
  72  *              WalkState           - Current method state
  73  *
  74  * RETURN:      Status
  75  *
  76  * DESCRIPTION: Convert Reference objects to values
  77  *
  78  ******************************************************************************/

  80 ACPI_STATUS
  81 AcpiExResolveToValue (
  82     ACPI_OPERAND_OBJECT     **StackPtr,
  83     ACPI_WALK_STATE         *WalkState)
  84 {
  85     ACPI_STATUS             Status;


  88     ACPI_FUNCTION_TRACE_PTR (ExResolveToValue, StackPtr);


  91     if (!StackPtr || !*StackPtr)
  92     {
  93         ACPI_ERROR ((AE_INFO, "Internal - null pointer"));
  94         return_ACPI_STATUS (AE_AML_NO_OPERAND);
  95     }

  97     /*
  98      * The entity pointed to by the StackPtr can be either
  99      * 1) A valid ACPI_OPERAND_OBJECT, or
 100      * 2) A ACPI_NAMESPACE_NODE (NamedObj)
 101      */
 102     if (ACPI_GET_DESCRIPTOR_TYPE (*StackPtr) == ACPI_DESC_TYPE_OPERAND)
 103     {
 104         Status = AcpiExResolveObjectToValue (StackPtr, WalkState);
 105         if (ACPI_FAILURE (Status))
 106         {
 107             return_ACPI_STATUS (Status);
 108         }

 110         if (!*StackPtr)
 111         {
 112             ACPI_ERROR ((AE_INFO, "Internal - null pointer"));
 113             return_ACPI_STATUS (AE_AML_NO_OPERAND);
 114         }
 115     }

 117     /*
 118      * Object on the stack may have changed if AcpiExResolveObjectToValue()
 119      * was called (i.e., we can't use an _else_ here.)
 120      */
 121     if (ACPI_GET_DESCRIPTOR_TYPE (*StackPtr) == ACPI_DESC_TYPE_NAMED)
 122     {
 123         Status = AcpiExResolveNodeToValue (
 124                         ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE, StackPtr),
```

```
125                         WalkState);
126         if (ACPI_FAILURE (Status))
127         {
128             return_ACPI_STATUS (Status);
129         }
130     }

132     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Resolved object %p\n", *StackPtr));
133     return_ACPI_STATUS (AE_OK);
134 }


137 /*******************************************************************************
138  *
139  * FUNCTION:    AcpiExResolveObjectToValue
140  *
141  * PARAMETERS:  StackPtr        - Pointer to an internal object
142  *              WalkState       - Current method state
143  *
144  * RETURN:      Status
145  *
146  * DESCRIPTION: Retrieve the value from an internal object. The Reference type
147  *              uses the associated AML opcode to determine the value.
148  *
149  ******************************************************************************/

151 static ACPI_STATUS
152 AcpiExResolveObjectToValue (
153     ACPI_OPERAND_OBJECT     **StackPtr,
154     ACPI_WALK_STATE         *WalkState)
155 {
156     ACPI_STATUS             Status = AE_OK;
157     ACPI_OPERAND_OBJECT     *StackDesc;
158     ACPI_OPERAND_OBJECT     *ObjDesc = NULL;
159     UINT8                   RefType;


162     ACPI_FUNCTION_TRACE (ExResolveObjectToValue);


165     StackDesc = *StackPtr;

167     /* This is an object of type ACPI_OPERAND_OBJECT */
168     /* This is an ACPI_OPERAND_OBJECT  */

169     switch (StackDesc->Common.Type)
170     {
171     case ACPI_TYPE_LOCAL_REFERENCE:

173         RefType = StackDesc->Reference.Class;

175         switch (RefType)
176         {
177         case ACPI_REFCLASS_LOCAL:
178         case ACPI_REFCLASS_ARG:

179             /*
180              * Get the local from the method's state info
181              * Note: this increments the local's object reference count
182              */
183             Status = AcpiDsMethodDataGetValue (RefType,
184                         StackDesc->Reference.Value, WalkState, &ObjDesc);
185             if (ACPI_FAILURE (Status))
186             {
187                 return_ACPI_STATUS (Status);
188             }
```



```
190             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "[Arg/Local %X] ValueObj is %p\n",
191                 StackDesc->Reference.Value, ObjDesc));

193             /*
194              * Now we can delete the original Reference Object and
195              * replace it with the resolved value
196              */
197             AcpiUtRemoveReference (StackDesc);
198             *StackPtr = ObjDesc;
199             break;


201         case ACPI_REFCLASS_INDEX:

203             switch (StackDesc->Reference.TargetType)
204             {
205             case ACPI_TYPE_BUFFER_FIELD:

207                 /* Just return - do not dereference */
208                 break;


210             case ACPI_TYPE_PACKAGE:

212                 /* If method call or CopyObject - do not dereference */

214                 if ((WalkState->Opcode == AML_INT_METHODCALL_OP) ||
215                     (WalkState->Opcode == AML_COPY_OP))
216                 {
217                     break;
218                 }

220                 /* Otherwise, dereference the PackageIndex to a package element

222                 ObjDesc = *StackDesc->Reference.Where;
223                 if (ObjDesc)
224                 {
225                     /*
226                      * Valid object descriptor, copy pointer to return value
227                      * (i.e., dereference the package index)
228                      * Delete the ref object, increment the returned object
229                      */
230                     AcpiUtRemoveReference (StackDesc);
231                     AcpiUtAddReference (ObjDesc);
232                     *StackPtr = ObjDesc;
233                 }
234                 else
235                 {
236                     /*
237                      * A NULL object descriptor means an uninitialized element o
238                      * the package, can't dereference it
239                      */
240                     ACPI_ERROR ((AE_INFO,
241                         "Attempt to dereference an Index to NULL package element
242                         StackDesc));
243                     Status = AE_AML_UNINITIALIZED_ELEMENT;
244                 }
245                 break;


247             default:

249                 /* Invalid reference object */

251                 ACPI_ERROR ((AE_INFO,
```

```
252                     "Unknown TargetType 0x%X in Index/Reference object %p",
253                     StackDesc->Reference.TargetType, StackDesc));
254                 Status = AE_AML_INTERNAL;
255                 break;
256             }
257             break;


259         case ACPI_REFCLASS_REFOF:
260         case ACPI_REFCLASS_DEBUG:
261         case ACPI_REFCLASS_TABLE:

263             /* Just leave the object as-is, do not dereference */

265             break;

267         case ACPI_REFCLASS_NAME:   /* Reference to a named object */

269             /* Dereference the name */

271             if ((StackDesc->Reference.Node->Type == ACPI_TYPE_DEVICE) ||
272                 (StackDesc->Reference.Node->Type == ACPI_TYPE_THERMAL))
273             {
274                 /* These node types do not have 'real' subobjects */

276                 *StackPtr = (void *) StackDesc->Reference.Node;
277             }
278             else
279             {
280                 /* Get the object pointed to by the namespace node */

282                 *StackPtr = (StackDesc->Reference.Node)->Object;
283                 AcpiUtAddReference (*StackPtr);
284             }

286             AcpiUtRemoveReference (StackDesc);
287             break;

289         default:

291             ACPI_ERROR ((AE_INFO,
292                 "Unknown Reference type 0x%X in %p", RefType, StackDesc));
293             Status = AE_AML_INTERNAL;
294             break;
295         }
296         break;


298     case ACPI_TYPE_BUFFER:

300         Status = AcpiDsGetBufferArguments (StackDesc);
301         break;


303     case ACPI_TYPE_PACKAGE:

305         Status = AcpiDsGetPackageArguments (StackDesc);
306         break;


308     case ACPI_TYPE_BUFFER_FIELD:
309     case ACPI_TYPE_LOCAL_REGION_FIELD:
310     case ACPI_TYPE_LOCAL_BANK_FIELD:
311     case ACPI_TYPE_LOCAL_INDEX_FIELD:

313         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "FieldRead SourceDesc=%p Type=%X\n",
```



```
314             StackDesc, StackDesc->Common.Type));

316         Status = AcpiExReadDataFromField (WalkState, StackDesc, &ObjDesc);

318         /* Remove a reference to the original operand, then override */

320         AcpiUtRemoveReference (*StackPtr);
321         *StackPtr = (void *) ObjDesc;
322         break;

324     default:

326         break;
327     }

329     return_ACPI_STATUS (Status);
330 }


333 /*******************************************************************************
334  *
335  * FUNCTION:    AcpiExResolveMultiple
336  *
337  * PARAMETERS:  WalkState           - Current state (contains AML opcode)
338  *              Operand             - Starting point for resolution
339  *              ReturnType          - Where the object type is returned
340  *              ReturnDesc          - Where the resolved object is returned
341  *
342  * RETURN:      Status
343  *
344  * DESCRIPTION: Return the base object and type. Traverse a reference list if
345  *              necessary to get to the base object.
346  *
347  ******************************************************************************/

349 ACPI_STATUS
350 AcpiExResolveMultiple (
351     ACPI_WALK_STATE         *WalkState,
352     ACPI_OPERAND_OBJECT     *Operand,
353     ACPI_OBJECT_TYPE        *ReturnType,
354     ACPI_OPERAND_OBJECT     **ReturnDesc)
355 {
356     ACPI_OPERAND_OBJECT     *ObjDesc = (void *) Operand;
357     ACPI_NAMESPACE_NODE     *Node;
358     ACPI_OBJECT_TYPE        Type;
359     ACPI_STATUS             Status;


362     ACPI_FUNCTION_TRACE (AcpiExResolveMultiple);


365     /* Operand can be either a namespace node or an operand descriptor */

367     switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
368     {
369     case ACPI_DESC_TYPE_OPERAND:

371         Type = ObjDesc->Common.Type;
372         break;

374     case ACPI_DESC_TYPE_NAMED:

376         Type = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;
377         ObjDesc = AcpiNsGetAttachedObject ((ACPI_NAMESPACE_NODE *) ObjDesc);

379         /* If we had an Alias node, use the attached object for type info */
```

```
381         if (Type == ACPI_TYPE_LOCAL_ALIAS)
382         {
383             Type = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;
384             ObjDesc = AcpiNsGetAttachedObject ((ACPI_NAMESPACE_NODE *) ObjDesc);
385         }
386         break;

388     default:
389         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
390     }

392     /* If type is anything other than a reference, we are done */

394     if (Type != ACPI_TYPE_LOCAL_REFERENCE)
395     {
396         goto Exit;
397     }

399     /*
400      * For reference objects created via the RefOf, Index, or Load/LoadTable
401      * operators, we need to get to the base object (as per the ACPI
402      * specification of the ObjectType and SizeOf operators). This means
403      * traversing the list of possibly many nested references.
404      */
405     while (ObjDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE)
406     {
407         switch (ObjDesc->Reference.Class)
408         {
409         case ACPI_REFCLASS_REFOF:
410         case ACPI_REFCLASS_NAME:

412             /* Dereference the reference pointer */

414             if (ObjDesc->Reference.Class == ACPI_REFCLASS_REFOF)
415             {
416                 Node = ObjDesc->Reference.Object;
417             }
418             else /* AML_INT_NAMEPATH_OP */
419             {
420                 Node = ObjDesc->Reference.Node;
421             }

423             /* All "References" point to a NS node */

425             if (ACPI_GET_DESCRIPTOR_TYPE (Node) != ACPI_DESC_TYPE_NAMED)
426             {
427                 ACPI_ERROR ((AE_INFO,
428                     "Not a namespace node %p [%s]",
429                     Node, AcpiUtGetDescriptorName (Node)));
430                 return_ACPI_STATUS (AE_AML_INTERNAL);
431             }

433             /* Get the attached object */

435             ObjDesc = AcpiNsGetAttachedObject (Node);
436             if (!ObjDesc)
437             {
438                 /* No object, use the NS node type */

440                 Type = AcpiNsGetType (Node);
441                 goto Exit;
442             }

444             /* Check for circular references */
```



```
446             if (ObjDesc == Operand)
447             {
448                 return_ACPI_STATUS (AE_AML_CIRCULAR_REFERENCE);
449             }
450             break;

452         case ACPI_REFCLASS_INDEX:

454             /* Get the type of this reference (index into another object) */

456             Type = ObjDesc->Reference.TargetType;
457             if (Type != ACPI_TYPE_PACKAGE)
458             {
459                 goto Exit;
460             }

462             /*
463              * The main object is a package, we want to get the type
464              * of the individual package element that is referenced by
465              * the index.
466              *
467              * This could of course in turn be another reference object.
468              */
469             ObjDesc = *(ObjDesc->Reference.Where);
470             if (!ObjDesc)
471             {
472                 /* NULL package elements are allowed */

474                 Type = 0; /* Uninitialized */
475                 goto Exit;
476             }
477             break;

479         case ACPI_REFCLASS_TABLE:

481             Type = ACPI_TYPE_DDB_HANDLE;
482             goto Exit;

484         case ACPI_REFCLASS_LOCAL:
485         case ACPI_REFCLASS_ARG:

487             if (ReturnDesc)
488             {
489                 Status = AcpiDsMethodDataGetValue (ObjDesc->Reference.Class,
490                             ObjDesc->Reference.Value, WalkState, &ObjDesc);
491                 if (ACPI_FAILURE (Status))
492                 {
493                     return_ACPI_STATUS (Status);
494                 }
495                 AcpiUtRemoveReference (ObjDesc);
496             }
497             else
498             {
499                 Status = AcpiDsMethodDataGetNode (ObjDesc->Reference.Class,
500                             ObjDesc->Reference.Value, WalkState, &Node);
501                 if (ACPI_FAILURE (Status))
502                 {
503                     return_ACPI_STATUS (Status);
504                 }

506                 ObjDesc = AcpiNsGetAttachedObject (Node);
507                 if (!ObjDesc)
508                 {
```

```
509                     Type = ACPI_TYPE_ANY;
510                     goto Exit;
511                 }
512             }
513             break;


515         case ACPI_REFCLASS_DEBUG:

517             /* The Debug Object is of type "DebugObject" */

519             Type = ACPI_TYPE_DEBUG_OBJECT;
520             goto Exit;


522         default:

524             ACPI_ERROR ((AE_INFO,
525                 "Unknown Reference Class 0x%2.2X", ObjDesc->Reference.Class));
526             return_ACPI_STATUS (AE_AML_INTERNAL);
527         }
528     }

530     /*
531      * Now we are guaranteed to have an object that has not been created
532      * via the RefOf or Index operators.
533      */
534     Type = ObjDesc->Common.Type;


537 Exit:
538     /* Convert internal types to external types */

540     switch (Type)
541     {
542     case ACPI_TYPE_LOCAL_REGION_FIELD:
543     case ACPI_TYPE_LOCAL_BANK_FIELD:
544     case ACPI_TYPE_LOCAL_INDEX_FIELD:

546         Type = ACPI_TYPE_FIELD_UNIT;
547         break;

549     case ACPI_TYPE_LOCAL_SCOPE:

551         /* Per ACPI Specification, Scope is untyped */

553         Type = ACPI_TYPE_ANY;
554         break;

556     default:

558         /* No change to Type required */

560         break;
561     }

563     *ReturnType = Type;
564     if (ReturnDesc)
565     {
566         *ReturnDesc = ObjDesc;
567     }
568     return_ACPI_STATUS (AE_OK);
569 }
```

```
**********************************************************
   22214 Thu Dec 26 13:49:09 2013
new/usr/src/common/acpica/components/executer/exresop.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exresop - AML Interpreter operand/object resolution
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXRESOP_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "amlcode.h"
  49 #include "acparser.h"
  50 #include "acinterp.h"
  51 #include "acnamesp.h"


  54 #define _COMPONENT          ACPI_EXECUTER
  55         ACPI_MODULE_NAME    ("exresop")

  57 /* Local prototypes */
```



```
  59 static ACPI_STATUS
  60 AcpiExCheckObjectType (
  61     ACPI_OBJECT_TYPE        TypeNeeded,
  62     ACPI_OBJECT_TYPE        ThisType,
  63     void                    *Object);


  66 /******************************************************************************
  67  *
  68  * FUNCTION:    AcpiExCheckObjectType
  69  *
  70  * PARAMETERS:  TypeNeeded          Object type needed
  71  *              ThisType            Actual object type
  72  *              Object              Object pointer
  73  *
  74  * RETURN:      Status
  75  *
  76  * DESCRIPTION: Check required type against actual type
  77  *
  78  *****************************************************************************/

  80 static ACPI_STATUS
  81 AcpiExCheckObjectType (
  82     ACPI_OBJECT_TYPE        TypeNeeded,
  83     ACPI_OBJECT_TYPE        ThisType,
  84     void                    *Object)
  85 {
  86     ACPI_FUNCTION_ENTRY ();


  89     if (TypeNeeded == ACPI_TYPE_ANY)
  90     {
  91         /* All types OK, so we don't perform any typechecks */

  93         return (AE_OK);
  94     }

  96     if (TypeNeeded == ACPI_TYPE_LOCAL_REFERENCE)
  97     {
  98         /*
  99          * Allow the AML "Constant" opcodes (Zero, One, etc.) to be reference
 100          * objects and thus allow them to be targets. (As per the ACPI
 101          * specification, a store to a constant is a noop.)
 102          */
 103         if ((ThisType == ACPI_TYPE_INTEGER) &&
 104             (((ACPI_OPERAND_OBJECT *) Object)->Common.Flags & AOPOBJ_AML_CONSTAN
 105         {
 106             return (AE_OK);
 107         }
 108     }

 110     if (TypeNeeded != ThisType)
 111     {
 112         ACPI_ERROR ((AE_INFO,
 113             "Needed type [%s], found [%s] %p",
 114             AcpiUtGetTypeName (TypeNeeded),
 115             AcpiUtGetTypeName (ThisType), Object));

 117         return (AE_AML_OPERAND_TYPE);
 118     }

 120     return (AE_OK);
 121 }


 124 /******************************************************************************
```

```
 *
 * FUNCTION:    AcpiExResolveOperands
 *
 * PARAMETERS:  Opcode              - Opcode being interpreted
 *              StackPtr            - Pointer to the operand stack to be
 *                                    resolved
 *              WalkState           - Current state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert multiple input operands to the types required by the
 *              target operator.
 *
 *      Each 5-bit group in ArgTypes represents one required
 *      operand and indicates the required Type. The corresponding operand
 *      will be converted to the required type if possible, otherwise we
 *      abort with an exception.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExResolveOperands (
    UINT16                  Opcode,
    ACPI_OPERAND_OBJECT     **StackPtr,
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_STATUS             Status = AE_OK;
    UINT8                   ObjectType;
    UINT32                  ArgTypes;
    const ACPI_OPCODE_INFO  *OpInfo;
    UINT32                  ThisArgType;
    ACPI_OBJECT_TYPE        TypeNeeded;
    UINT16                  TargetOp = 0;


    ACPI_FUNCTION_TRACE_U32 (ExResolveOperands, Opcode);


    OpInfo = AcpiPsGetOpcodeInfo (Opcode);
    if (OpInfo->Class == AML_CLASS_UNKNOWN)
    {
        return_ACPI_STATUS (AE_AML_BAD_OPCODE);
    }

    ArgTypes = OpInfo->RuntimeArgs;
    if (ArgTypes == ARGI_INVALID_OPCODE)
    {
        ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
            Opcode));

        return_ACPI_STATUS (AE_AML_INTERNAL);
    }

    ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
        "Opcode %X [%s] RequiredOperandTypes=%8.8X\n",
        Opcode, OpInfo->Name, ArgTypes));

    /*
     * Normal exit is with (ArgTypes == 0) at end of argument list.
     * Function will return an exception from within the loop upon
     * finding an entry which is not (or cannot be converted
     * to) the required type; if stack underflows; or upon
     * finding a NULL stack entry (which should not happen).
     */
    while (GET_CURRENT_ARG_TYPE (ArgTypes))
```



```
    {
        if (!StackPtr || !*StackPtr)
        {
            ACPI_ERROR ((AE_INFO, "Null stack entry at %p",
                StackPtr));

            return_ACPI_STATUS (AE_AML_INTERNAL);
        }

        /* Extract useful items */

        ObjDesc = *StackPtr;

        /* Decode the descriptor type */

        switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
        {
        case ACPI_DESC_TYPE_NAMED:

            /* Namespace Node */

            ObjectType = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;

            /*
             * Resolve an alias object. The construction of these objects
             * guarantees that there is only one level of alias indirection;
             * thus, the attached object is always the aliased namespace node
             */
            if (ObjectType == ACPI_TYPE_LOCAL_ALIAS)
            {
                ObjDesc = AcpiNsGetAttachedObject ((ACPI_NAMESPACE_NODE *) ObjDe
                *StackPtr = ObjDesc;
                ObjectType = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;
            }
            break;


        case ACPI_DESC_TYPE_OPERAND:

            /* ACPI internal object */

            ObjectType = ObjDesc->Common.Type;

            /* Check for bad ACPI_OBJECT_TYPE */

            if (!AcpiUtValidObjectType (ObjectType))
            {
                ACPI_ERROR ((AE_INFO,
                    "Bad operand object type [0x%X]", ObjectType));

                return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
            }

            if (ObjectType == (UINT8) ACPI_TYPE_LOCAL_REFERENCE)
            {
                /* Validate the Reference */

                switch (ObjDesc->Reference.Class)
                {
                case ACPI_REFCLASS_DEBUG:

                    TargetOp = AML_DEBUG_OP;

                    /*lint -fallthrough */

                case ACPI_REFCLASS_ARG:
```

```
256                 case ACPI_REFCLASS_LOCAL:
257                 case ACPI_REFCLASS_INDEX:
258                 case ACPI_REFCLASS_REFOF:
259                 case ACPI_REFCLASS_TABLE:    /* DdbHandle from LOAD_OP or LOAD_T
260                 case ACPI_REFCLASS_NAME:     /* Reference to a named object */

262                     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
263                         "Operand is a Reference, Class [%s] %2.2X\n",
264                         AcpiUtGetReferenceName (ObjDesc),
265                         ObjDesc->Reference.Class));
266                     break;

268                 default:

270                     ACPI_ERROR ((AE_INFO,
271                         "Unknown Reference Class 0x%2.2X in %p",
272                         ObjDesc->Reference.Class, ObjDesc));

274                     return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
275                 }
276             }
277             break;


279         default:

281             /* Invalid descriptor */

283             ACPI_ERROR ((AE_INFO, "Invalid descriptor %p [%s]",
284                 ObjDesc, AcpiUtGetDescriptorName (ObjDesc)));

286             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
287         }

289         /* Get one argument type, point to the next */

291         ThisArgType = GET_CURRENT_ARG_TYPE (ArgTypes);
292         INCREMENT_ARG_LIST (ArgTypes);

294         /*
295          * Handle cases where the object does not need to be
296          * resolved to a value
297          */
298         switch (ThisArgType)
299         {
300         case ARGI_REF_OR_STRING:        /* Can be a String or Reference */

302             if ((ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) == ACPI_DESC_TYPE_OPERAND) &
303                 (ObjDesc->Common.Type == ACPI_TYPE_STRING))
304             {
305                 /*
306                  * String found - the string references a named object and
307                  * must be resolved to a node
308                  */
309                 goto NextOperand;
310             }

312             /*
313              * Else not a string - fall through to the normal Reference
314              * case below
315              */
316             /*lint -fallthrough */

318         case ARGI_REFERENCE:            /* References: */
319         case ARGI_INTEGER_REF:
320         case ARGI_OBJECT_REF:
```



```
321         case ARGI_DEVICE_REF:
322         case ARGI_TARGETREF:     /* Allows implicit conversion rules before stor
323         case ARGI_FIXED_TARGET:  /* No implicit conversion before store to targe
324         case ARGI_SIMPLE_TARGET: /* Name, Local, or Arg - no implicit conversion

325             /*
326              * Need an operand of type ACPI_TYPE_LOCAL_REFERENCE
327              * A Namespace Node is OK as-is
328              */
329             if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) == ACPI_DESC_TYPE_NAMED)
330             {
331                 goto NextOperand;
332             }

334             Status = AcpiExCheckObjectType (ACPI_TYPE_LOCAL_REFERENCE,
335                             ObjectType, ObjDesc);
336             if (ACPI_FAILURE (Status))
337             {
338                 return_ACPI_STATUS (Status);
339             }
340             goto NextOperand;


342         case ARGI_DATAREFOBJ:  /* Store operator only */

343             /*
344              * We don't want to resolve IndexOp reference objects during
345              * a store because this would be an implicit DeRefOf operation.
346              * Instead, we just want to store the reference object.
347              * -- All others must be resolved below.
348              */
349             if ((Opcode == AML_STORE_OP) &&
350                 ((*StackPtr)->Common.Type == ACPI_TYPE_LOCAL_REFERENCE) &&
351                 ((*StackPtr)->Reference.Class == ACPI_REFCLASS_INDEX))
352             {
353                 goto NextOperand;
354             }
355             break;

357         default:

359             /* All cases covered above */

361             break;
362         }

364         /*
365          * Resolve this object to a value
366          */
367         Status = AcpiExResolveToValue (StackPtr, WalkState);
368         if (ACPI_FAILURE (Status))
369         {
370             return_ACPI_STATUS (Status);
371         }

373         /* Get the resolved object */

375         ObjDesc = *StackPtr;

377         /*
378          * Check the resulting object (value) type
379          */
380         switch (ThisArgType)
381         {
382         /*
383          * For the simple cases, only one type of resolved object
```

```
384          * is allowed
385          */
386         case ARGI_MUTEX:

388             /* Need an operand of type ACPI_TYPE_MUTEX */

390             TypeNeeded = ACPI_TYPE_MUTEX;
391             break;

393         case ARGI_EVENT:

395             /* Need an operand of type ACPI_TYPE_EVENT */

397             TypeNeeded = ACPI_TYPE_EVENT;
398             break;

400         case ARGI_PACKAGE:   /* Package */

402             /* Need an operand of type ACPI_TYPE_PACKAGE */

404             TypeNeeded = ACPI_TYPE_PACKAGE;
405             break;

407         case ARGI_ANYTYPE:

409             /* Any operand type will do */

411             TypeNeeded = ACPI_TYPE_ANY;
412             break;

414         case ARGI_DDBHANDLE:

416             /* Need an operand of type ACPI_TYPE_DDB_HANDLE */

418             TypeNeeded = ACPI_TYPE_LOCAL_REFERENCE;
419             break;


422         /*
423          * The more complex cases allow multiple resolved object types
424          */
425         case ARGI_INTEGER:

427             /*
428              * Need an operand of type ACPI_TYPE_INTEGER,
429              * But we can implicitly convert from a STRING or BUFFER
430              * Aka - "Implicit Source Operand Conversion"
431              */
432             Status = AcpiExConvertToInteger (ObjDesc, StackPtr, 16);
433             if (ACPI_FAILURE (Status))
434             {
435                 if (Status == AE_TYPE)
436                 {
437                     ACPI_ERROR ((AE_INFO,
438                         "Needed [Integer/String/Buffer], found [%s] %p",
439                         AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

441                     return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
442                 }

444                 return_ACPI_STATUS (Status);
445             }

447             if (ObjDesc != *StackPtr)
448             {
449                 AcpiUtRemoveReference (ObjDesc);
```



```
450             }
451             goto NextOperand;


453         case ARGI_BUFFER:

454             /*
455              * Need an operand of type ACPI_TYPE_BUFFER,
456              * But we can implicitly convert from a STRING or INTEGER
457              * Aka - "Implicit Source Operand Conversion"
458              */
459             Status = AcpiExConvertToBuffer (ObjDesc, StackPtr);
460             if (ACPI_FAILURE (Status))
461             {
462                 if (Status == AE_TYPE)
463                 {
464                     ACPI_ERROR ((AE_INFO,
465                         "Needed [Integer/String/Buffer], found [%s] %p",
466                         AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

468                     return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
469                 }

471                 return_ACPI_STATUS (Status);
472             }

474             if (ObjDesc != *StackPtr)
475             {
476                 AcpiUtRemoveReference (ObjDesc);
477             }
478             goto NextOperand;


480         case ARGI_STRING:

481             /*
482              * Need an operand of type ACPI_TYPE_STRING,
483              * But we can implicitly convert from a BUFFER or INTEGER
484              * Aka - "Implicit Source Operand Conversion"
485              */
486             Status = AcpiExConvertToString (ObjDesc, StackPtr,
487                         ACPI_IMPLICIT_CONVERT_HEX);
488             if (ACPI_FAILURE (Status))
489             {
490                 if (Status == AE_TYPE)
491                 {
492                     ACPI_ERROR ((AE_INFO,
493                         "Needed [Integer/String/Buffer], found [%s] %p",
494                         AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

496                     return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
497                 }

499                 return_ACPI_STATUS (Status);
500             }

502             if (ObjDesc != *StackPtr)
503             {
504                 AcpiUtRemoveReference (ObjDesc);
505             }
506             goto NextOperand;


508         case ARGI_COMPUTEDATA:

510             /* Need an operand of type INTEGER, STRING or BUFFER */
```

```
512             switch (ObjDesc->Common.Type)
513             {
514             case ACPI_TYPE_INTEGER:
515             case ACPI_TYPE_STRING:
516             case ACPI_TYPE_BUFFER:

518                 /* Valid operand */
519                break;

521             default:
522                 ACPI_ERROR ((AE_INFO,
523                     "Needed [Integer/String/Buffer], found [%s] %p",
524                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

526                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
527             }
528             goto NextOperand;


530         case ARGI_BUFFER_OR_STRING:

532             /* Need an operand of type STRING or BUFFER */

534             switch (ObjDesc->Common.Type)
535             {
536             case ACPI_TYPE_STRING:
537             case ACPI_TYPE_BUFFER:

539                 /* Valid operand */
540                break;

542             case ACPI_TYPE_INTEGER:

544                 /* Highest priority conversion is to type Buffer */

546                 Status = AcpiExConvertToBuffer (ObjDesc, StackPtr);
547                 if (ACPI_FAILURE (Status))
548                 {
549                     return_ACPI_STATUS (Status);
550                 }

552                 if (ObjDesc != *StackPtr)
553                 {
554                     AcpiUtRemoveReference (ObjDesc);
555                 }
556                 break;

558             default:
559                 ACPI_ERROR ((AE_INFO,
560                     "Needed [Integer/String/Buffer], found [%s] %p",
561                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

563                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
564             }
565             goto NextOperand;


567         case ARGI_DATAOBJECT:
568             /*
569              * ARGI_DATAOBJECT is only used by the SizeOf operator.
570              * Need a buffer, string, package, or RefOf reference.
571              *
572              * The only reference allowed here is a direct reference to
573              * a namespace node.
574              */
```



```
575             switch (ObjDesc->Common.Type)
576             {
577             case ACPI_TYPE_PACKAGE:
578             case ACPI_TYPE_STRING:
579             case ACPI_TYPE_BUFFER:
580             case ACPI_TYPE_LOCAL_REFERENCE:

582                 /* Valid operand */
583                 break;

585             default:

587                 ACPI_ERROR ((AE_INFO,
588                     "Needed [Buffer/String/Package/Reference], found [%s] %p",
589                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

591                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
592             }
593             goto NextOperand;


595         case ARGI_COMPLEXOBJ:

597             /* Need a buffer or package or (ACPI 2.0) String */

599             switch (ObjDesc->Common.Type)
600             {
601             case ACPI_TYPE_PACKAGE:
602             case ACPI_TYPE_STRING:
603             case ACPI_TYPE_BUFFER:

605                 /* Valid operand */
606                 break;

608             default:

610                 ACPI_ERROR ((AE_INFO,
611                     "Needed [Buffer/String/Package], found [%s] %p",
612                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

614                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
615             }
616             goto NextOperand;


618         case ARGI_REGION_OR_BUFFER: /* Used by Load() only */

620             /* Need an operand of type REGION or a BUFFER (which could be a reso

622             switch (ObjDesc->Common.Type)
623             {
624             case ACPI_TYPE_BUFFER:
625             case ACPI_TYPE_REGION:

627                 /* Valid operand */
628                 break;

630             default:

632                 ACPI_ERROR ((AE_INFO,
633                     "Needed [Region/Buffer], found [%s] %p",
634                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

636                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
637             }
638             goto NextOperand;
```

```
640         case ARGI_DATAREFOBJ:

642             /* Used by the Store() operator only */

644             switch (ObjDesc->Common.Type)
645             {
646             case ACPI_TYPE_INTEGER:
647             case ACPI_TYPE_PACKAGE:
648             case ACPI_TYPE_STRING:
649             case ACPI_TYPE_BUFFER:
650             case ACPI_TYPE_BUFFER_FIELD:
651             case ACPI_TYPE_LOCAL_REFERENCE:
652             case ACPI_TYPE_LOCAL_REGION_FIELD:
653             case ACPI_TYPE_LOCAL_BANK_FIELD:
654             case ACPI_TYPE_LOCAL_INDEX_FIELD:
655             case ACPI_TYPE_DDB_HANDLE:

657                 /* Valid operand */
658                 break;

660             default:

662                 if (AcpiGbl_EnableInterpreterSlack)
663                 {
664                     /*
665                      * Enable original behavior of Store(), allowing any and all
666                      * objects as the source operand. The ACPI spec does not
667                      * allow this, however.
668                      */
669                     break;
670                 }

672                 if (TargetOp == AML_DEBUG_OP)
673                 {
674                     /* Allow store of any object to the Debug object */

676                     break;
677                 }

679                 ACPI_ERROR ((AE_INFO,
680                     "Needed Integer/Buffer/String/Package/Ref/Ddb], found [%s] %
681                     AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));

683                 return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
684             }
685             goto NextOperand;


687         default:

689             /* Unknown type */

691             ACPI_ERROR ((AE_INFO,
692                 "Internal - Unknown ARGI (required operand) type 0x%X",
693                 ThisArgType));

695             return_ACPI_STATUS (AE_BAD_PARAMETER);
696         }

698         /*
699          * Make sure that the original object was resolved to the
700          * required object type (Simple cases only).
701          */
702         Status = AcpiExCheckObjectType (TypeNeeded,
```



```
703                         (*StackPtr)->Common.Type, *StackPtr);
704         if (ACPI_FAILURE (Status))
705         {
706             return_ACPI_STATUS (Status);
707         }

709 NextOperand:
710         /*
711          * If more operands needed, decrement StackPtr to point
712          * to next operand on stack
713          */
714         if (GET_CURRENT_ARG_TYPE (ArgTypes))
715         {
716             StackPtr--;
717         }
718     }

720     ACPI_DUMP_OPERANDS (WalkState->Operands,
721         AcpiPsGetOpcodeName (Opcode), WalkState->NumOperands);

723     return_ACPI_STATUS (Status);
724 }
```

```
**********************************************************
   18686 Thu Dec 26 13:49:09 2013
new/usr/src/common/acpica/components/executer/exstore.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: exstore - AML Interpreter object store support
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __EXSTORE_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acdispat.h"
  49 #include "acinterp.h"
  50 #include "amlcode.h"
  51 #include "acnamesp.h"


  54 #define _COMPONENT          ACPI_EXECUTER
  55         ACPI_MODULE_NAME    ("exstore")

  57 /* Local prototypes */
```



```
  59 static ACPI_STATUS
  60 AcpiExStoreObjectToIndex (
  61     ACPI_OPERAND_OBJECT     *ValDesc,
  62     ACPI_OPERAND_OBJECT     *DestDesc,
  63     ACPI_WALK_STATE         *WalkState);

  65 static ACPI_STATUS
  66 AcpiExStoreDirectToNode (
  67     ACPI_OPERAND_OBJECT     *SourceDesc,
  68     ACPI_NAMESPACE_NODE     *Node,
  69     ACPI_WALK_STATE         *WalkState);


  72 /*******************************************************************************
  73  *
  74  * FUNCTION:    AcpiExStore
  75  *
  76  * PARAMETERS:  *SourceDesc         - Value to be stored
  77  *              *DestDesc           - Where to store it. Must be an NS node
  78  *                                    or ACPI_OPERAND_OBJECT of type
  72  *                                    or an ACPI_OPERAND_OBJECT of type
  79  *                                    Reference;
  80  *              WalkState           - Current walk state
  81  *
  82  * RETURN:      Status
  83  *
  84  * DESCRIPTION: Store the value described by SourceDesc into the location
  85  *              described by DestDesc. Called by various interpreter
  86  *              functions to store the result of an operation into
  87  *              the destination operand -- not just simply the actual "Store"
  88  *              ASL operator.
  89  *
  90  ******************************************************************************/

  92 ACPI_STATUS
  93 AcpiExStore (
  94     ACPI_OPERAND_OBJECT     *SourceDesc,
  95     ACPI_OPERAND_OBJECT     *DestDesc,
  96     ACPI_WALK_STATE         *WalkState)
  97 {
  98     ACPI_STATUS             Status = AE_OK;
  99     ACPI_OPERAND_OBJECT     *RefDesc = DestDesc;


 102     ACPI_FUNCTION_TRACE_PTR (ExStore, DestDesc);


 105     /* Validate parameters */

 107     if (!SourceDesc || !DestDesc)
 108     {
 109         ACPI_ERROR ((AE_INFO, "Null parameter"));
 110         return_ACPI_STATUS (AE_AML_NO_OPERAND);
 111     }

 113     /* DestDesc can be either a namespace node or an ACPI object */

 115     if (ACPI_GET_DESCRIPTOR_TYPE (DestDesc) == ACPI_DESC_TYPE_NAMED)
 116     {
 117         /*
 118          * Dest is a namespace node,
 119          * Storing an object into a Named node.
 120          */
 121         Status = AcpiExStoreObjectToNode (SourceDesc,
 122                     (ACPI_NAMESPACE_NODE *) DestDesc, WalkState,
 123                     ACPI_IMPLICIT_CONVERSION);
```

```
125         return_ACPI_STATUS (Status);
126     }

128     /* Destination object must be a Reference or a Constant object */

130     switch (DestDesc->Common.Type)
131     {
132     case ACPI_TYPE_LOCAL_REFERENCE:

134         break;

136     case ACPI_TYPE_INTEGER:

138         /* Allow stores to Constants -- a Noop as per ACPI spec */

140         if (DestDesc->Common.Flags & AOPOBJ_AML_CONSTANT)
141         {
142             return_ACPI_STATUS (AE_OK);
143         }

145         /*lint -fallthrough */

147     default:

149         /* Destination is not a Reference object */

151         ACPI_ERROR ((AE_INFO,
152             "Target is not a Reference or Constant object - %s [%p]",
153             AcpiUtGetObjectTypeName (DestDesc), DestDesc));

155         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
156     }

158     /*
159      * Examine the Reference class. These cases are handled:
160      *
161      * 1) Store to Name (Change the object associated with a name)
162      * 2) Store to an indexed area of a Buffer or Package
163      * 3) Store to a Method Local or Arg
164      * 4) Store to the debug object
165      */
166     switch (RefDesc->Reference.Class)
167     {
168     case ACPI_REFCLASS_REFOF:

170         /* Storing an object into a Name "container" */

172         Status = AcpiExStoreObjectToNode (SourceDesc,
173                     RefDesc->Reference.Object,
174                     WalkState, ACPI_IMPLICIT_CONVERSION);
175         break;


177     case ACPI_REFCLASS_INDEX:

179         /* Storing to an Index (pointer into a packager or buffer) */

181         Status = AcpiExStoreObjectToIndex (SourceDesc, RefDesc, WalkState);
182         break;


184     case ACPI_REFCLASS_LOCAL:
185     case ACPI_REFCLASS_ARG:

187         /* Store to a method local/arg  */
```



```
189         Status = AcpiDsStoreObjectToLocal (RefDesc->Reference.Class,
190                     RefDesc->Reference.Value, SourceDesc, WalkState);
191         break;


193     case ACPI_REFCLASS_DEBUG:

194         /*
195          * Storing to the Debug object causes the value stored to be
196          * displayed and otherwise has no effect -- see ACPI Specification
197          */
198         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
199             "**** Write to Debug Object: Object %p %s ****:\n\n",
200             SourceDesc, AcpiUtGetObjectTypeName (SourceDesc)));

202         ACPI_DEBUG_OBJECT (SourceDesc, 0, 0);
203         break;


205     default:

207         ACPI_ERROR ((AE_INFO, "Unknown Reference Class 0x%2.2X",
208             RefDesc->Reference.Class));
209         ACPI_DUMP_ENTRY (RefDesc, ACPI_LV_INFO);

211         Status = AE_AML_INTERNAL;
212         break;
213     }

215     return_ACPI_STATUS (Status);
216 }


219 /*******************************************************************************
220  *
221  * FUNCTION:    AcpiExStoreObjectToIndex
222  *
223  * PARAMETERS:  *SourceDesc             - Value to be stored
224  *              *DestDesc               - Named object to receive the value
225  *              WalkState               - Current walk state
226  *
227  * RETURN:      Status
228  *
229  * DESCRIPTION: Store the object to indexed Buffer or Package element
230  *
231  ******************************************************************************/

233 static ACPI_STATUS
234 AcpiExStoreObjectToIndex (
235     ACPI_OPERAND_OBJECT     *SourceDesc,
236     ACPI_OPERAND_OBJECT     *IndexDesc,
237     ACPI_WALK_STATE         *WalkState)
238 {
239     ACPI_STATUS             Status = AE_OK;
240     ACPI_OPERAND_OBJECT     *ObjDesc;
241     ACPI_OPERAND_OBJECT     *NewDesc;
242     UINT8                   Value = 0;
243     UINT32                  i;


246     ACPI_FUNCTION_TRACE (ExStoreObjectToIndex);


249     /*
250      * Destination must be a reference pointer, and
```

```
251      * must point to either a buffer or a package
252      */
253     switch (IndexDesc->Reference.TargetType)
254     {
255     case ACPI_TYPE_PACKAGE:
256         /*
257          * Storing to a package element. Copy the object and replace
258          * any existing object with the new object. No implicit
259          * conversion is performed.
260          *
261          * The object at *(IndexDesc->Reference.Where) is the
262          * element within the package that is to be modified.
263          * The parent package object is at IndexDesc->Reference.Object
264          */
265         ObjDesc = *(IndexDesc->Reference.Where);

267         if (SourceDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE &&
268             SourceDesc->Reference.Class == ACPI_REFCLASS_TABLE)
269         {
270             /* This is a DDBHandle, just add a reference to it */

272             AcpiUtAddReference (SourceDesc);
273             NewDesc = SourceDesc;
274         }
275         else
276         {
277             /* Normal object, copy it */

279             Status = AcpiUtCopyIobjectToIobject (SourceDesc, &NewDesc, WalkState
280             if (ACPI_FAILURE (Status))
281             {
282                 return_ACPI_STATUS (Status);
283             }
284         }

286         if (ObjDesc)
287         {
288             /* Decrement reference count by the ref count of the parent package

290             for (i = 0;
291                  i < ((ACPI_OPERAND_OBJECT *)
292                       IndexDesc->Reference.Object)->Common.ReferenceCount;
293                  i++)
294             {
295                 AcpiUtRemoveReference (ObjDesc);
296             }
297         }

299         *(IndexDesc->Reference.Where) = NewDesc;

301         /* Increment ref count by the ref count of the parent package-1 */

303         for (i = 1;
304              i < ((ACPI_OPERAND_OBJECT *)
305                   IndexDesc->Reference.Object)->Common.ReferenceCount;
306              i++)
307         {
308             AcpiUtAddReference (NewDesc);
309         }

311         break;


313     case ACPI_TYPE_BUFFER_FIELD:

314         /*
```



```
315          * Store into a Buffer or String (not actually a real BufferField)
316          * at a location defined by an Index.
317          *
318          * The first 8-bit element of the source object is written to the
319          * 8-bit Buffer location defined by the Index destination object,
320          * according to the ACPI 2.0 specification.
321          */

323         /*
324          * Make sure the target is a Buffer or String. An error should
325          * not happen here, since the ReferenceObject was constructed
326          * by the INDEX_OP code.
327          */
328         ObjDesc = IndexDesc->Reference.Object;
329         if ((ObjDesc->Common.Type != ACPI_TYPE_BUFFER) &&
330             (ObjDesc->Common.Type != ACPI_TYPE_STRING))
331         {
332             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
333         }

335         /*
336          * The assignment of the individual elements will be slightly
337          * different for each source type.
338          */
339         switch (SourceDesc->Common.Type)
340         {
341         case ACPI_TYPE_INTEGER:

343             /* Use the least-significant byte of the integer */

345             Value = (UINT8) (SourceDesc->Integer.Value);
346             break;

348         case ACPI_TYPE_BUFFER:
349         case ACPI_TYPE_STRING:

351             /* Note: Takes advantage of common string/buffer fields */

353             Value = SourceDesc->Buffer.Pointer[0];
354             break;

356         default:

358             /* All other types are invalid */

360             ACPI_ERROR ((AE_INFO,
361                 "Source must be Integer/Buffer/String type, not %s",
362                 AcpiUtGetObjectTypeName (SourceDesc)));
363             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
364         }

366         /* Store the source value into the target buffer byte */

368         ObjDesc->Buffer.Pointer[IndexDesc->Reference.Value] = Value;
369         break;


371     default:
372         ACPI_ERROR ((AE_INFO,
373             "Target is not a Package or BufferField"));
374         Status = AE_AML_OPERAND_TYPE;
375         break;
376     }

378     return_ACPI_STATUS (Status);
379 }
```

```
382 /*******************************************************************************
383  *
384  * FUNCTION:    AcpiExStoreObjectToNode
385  *
386  * PARAMETERS:  SourceDesc              - Value to be stored
387  *              Node                    - Named object to receive the value
388  *              WalkState               - Current walk state
389  *              ImplicitConversion      - Perform implicit conversion (yes/no)
390  *
391  * RETURN:      Status
392  *
393  * DESCRIPTION: Store the object to the named object.
394  *
395  *              The Assignment of an object to a named object is handled here
396  *              The value passed in will replace the current value (if any)
397  *              with the input value.
398  *
399  *              When storing into an object the data is converted to the
400  *              target object type then stored in the object. This means
401  *              that the target object type (for an initialized target) will
402  *              not be changed by a store operation. A CopyObject can change
403  *              the target type, however.
403  *              not be changed by a store operation.
404  *
405  *              The ImplicitConversion flag is set to NO/FALSE only when
406  *              storing to an ArgX -- as per the rules of the ACPI spec.
407  *
408  *              Assumes parameters are already validated.
409  *
410  ******************************************************************************/

412 ACPI_STATUS
413 AcpiExStoreObjectToNode (
414     ACPI_OPERAND_OBJECT     *SourceDesc,
415     ACPI_NAMESPACE_NODE     *Node,
416     ACPI_WALK_STATE         *WalkState,
417     UINT8                   ImplicitConversion)
418 {
419     ACPI_STATUS             Status = AE_OK;
420     ACPI_OPERAND_OBJECT     *TargetDesc;
421     ACPI_OPERAND_OBJECT     *NewDesc;
422     ACPI_OBJECT_TYPE        TargetType;


425     ACPI_FUNCTION_TRACE_PTR (ExStoreObjectToNode, SourceDesc);


428     /* Get current type of the node, and object attached to Node */

430     TargetType = AcpiNsGetType (Node);
431     TargetDesc = AcpiNsGetAttachedObject (Node);

433     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Storing %p (%s) to node %p (%s)\n",
430     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Storing %p(%s) into node %p(%s)\n",
434         SourceDesc, AcpiUtGetObjectTypeName (SourceDesc),
435               Node, AcpiUtGetTypeName (TargetType)));

437     /*
438      * Resolve the source object to an actual value
439      * (If it is a reference object)
440      */
441     Status = AcpiExResolveObject (&SourceDesc, TargetType, WalkState);
442     if (ACPI_FAILURE (Status))
443     {
```



```
444         return_ACPI_STATUS (Status);
445     }

444     /* If no implicit conversion, drop into the default case below */

446     if ((!ImplicitConversion) ||
447         ((WalkState->Opcode == AML_COPY_OP) &&
448          (TargetType != ACPI_TYPE_LOCAL_REGION_FIELD) &&
449          (TargetType != ACPI_TYPE_LOCAL_BANK_FIELD) &&
450          (TargetType != ACPI_TYPE_LOCAL_INDEX_FIELD)))
451     {
452         /*
453          * Force execution of default (no implicit conversion). Note:
454          * CopyObject does not perform an implicit conversion, as per the ACPI
455          * spec -- except in case of region/bank/index fields -- because these
456          * objects must retain their original type permanently.
457          */
458         TargetType = ACPI_TYPE_ANY;
459     }

447     /* Do the actual store operation */

449     switch (TargetType)
450     {
465     case ACPI_TYPE_BUFFER_FIELD:
466     case ACPI_TYPE_LOCAL_REGION_FIELD:
467     case ACPI_TYPE_LOCAL_BANK_FIELD:
468     case ACPI_TYPE_LOCAL_INDEX_FIELD:

470         /* For fields, copy the source data to the target field. */

472         Status = AcpiExWriteDataToField (SourceDesc, TargetDesc,
473                     &WalkState->ResultObj);
474         break;


451     case ACPI_TYPE_INTEGER:
452     case ACPI_TYPE_STRING:
453     case ACPI_TYPE_BUFFER:
454         /*
455          * The simple data types all support implicit source operand
456          * conversion before the store.
457          */

459         if ((WalkState->Opcode == AML_COPY_OP) ||
460             !ImplicitConversion)
461         {
462             /*
463              * However, CopyObject and Stores to ArgX do not perform
464              * an implicit conversion, as per the ACPI specification.
465              * A direct store is performed instead.
482          * These target types are all of type Integer/String/Buffer, and
483          * therefore support implicit conversion before the store.
484          *
485          * Copy and/or convert the source object to a new target object
466              */
467             Status = AcpiExStoreDirectToNode (SourceDesc, Node,
468                 WalkState);
469             break;
470         }

472         /* Store with implicit source operand conversion support */

474         Status = AcpiExStoreObjectToObject (SourceDesc, TargetDesc,
475             &NewDesc, WalkState);
476         if (ACPI_FAILURE (Status))
```

```
        {
            return_ACPI_STATUS (Status);
        }

        if (NewDesc != TargetDesc)
        {
            /*
             * Store the new NewDesc as the new value of the Name, and set
             * the Name's type to that of the value being stored in it.
             * SourceDesc reference count is incremented by AttachObject.
             *
             * Note: This may change the type of the node if an explicit
             * store has been performed such that the node/object type
             * has been changed.
             * Note: This may change the type of the node if an explicit store
             * has been performed such that the node/object type has been
             * changed.
             */
            Status = AcpiNsAttachObject (Node, NewDesc,
                NewDesc->Common.Type);
            Status = AcpiNsAttachObject (Node, NewDesc, NewDesc->Common.Type);

            ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
                "Store %s into %s via Convert/Attach\n",
                AcpiUtGetObjectTypeName (SourceDesc),
                AcpiUtGetObjectTypeName (NewDesc)));
        }
        break;

    case ACPI_TYPE_BUFFER_FIELD:
    case ACPI_TYPE_LOCAL_REGION_FIELD:
    case ACPI_TYPE_LOCAL_BANK_FIELD:
    case ACPI_TYPE_LOCAL_INDEX_FIELD:
        /*
         * For all fields, always write the source data to the target
         * field. Any required implicit source operand conversion is
         * performed in the function below as necessary. Note, field
         * objects must retain their original type permanently.
         */
        Status = AcpiExWriteDataToField (SourceDesc, TargetDesc,
            &WalkState->ResultObj);
        break;

    default:
        /*
         * No conversions for all other types. Directly store a copy of
         * the source object. This is the ACPI spec-defined behavior for
         * the CopyObject operator.
         *
         * NOTE: For the Store operator, this is a departure from the
         * ACPI spec, which states "If conversion is impossible, abort
         * the running control method". Instead, this code implements
         * "If conversion is impossible, treat the Store operation as
         * a CopyObject".
         */
        Status = AcpiExStoreDirectToNode (SourceDesc, Node,
            WalkState);
        break;
    }

    return_ACPI_STATUS (Status);
}


/*****************************************************************************
 *
```



```
 * FUNCTION:    AcpiExStoreDirectToNode
 *
 * PARAMETERS:  SourceDesc              - Value to be stored
 *              Node                    - Named object to receive the value
 *              WalkState               - Current walk state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: "Store" an object directly to a node. This involves a copy
 *              and an attach.
 *
 ****************************************************************************/

static ACPI_STATUS
AcpiExStoreDirectToNode (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *NewDesc;


    ACPI_FUNCTION_TRACE (ExStoreDirectToNode);


    ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
        "Storing [%s] (%p) directly into node [%s] (%p)"
        " with no implicit conversion\n",
        AcpiUtGetObjectTypeName (SourceDesc), SourceDesc,
        AcpiUtGetTypeName (Node->Type), Node));
            "Storing %s (%p) directly into node (%p) with no implicit conversion
           AcpiUtGetObjectTypeName (SourceDesc), SourceDesc, Node));

    /* Copy the source object to a new object */
        /* No conversions for all other types.  Just attach the source object */

    Status = AcpiUtCopyIobjectToIobject (SourceDesc, &NewDesc, WalkState);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
        Status = AcpiNsAttachObject (Node, SourceDesc,
                    SourceDesc->Common.Type);
        break;
    }

    /* Attach the new object to the node */

    Status = AcpiNsAttachObject (Node, NewDesc, NewDesc->Common.Type);
    AcpiUtRemoveReference (NewDesc);
    return_ACPI_STATUS (Status);
}
```

****
10281 Thu Dec 26 13:49:10 2013
new/usr/src/common/acpica/components/executer/exstoren.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exstoren - AML Interpreter object store support,
 *                        Store to Node (namespace object)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXSTOREN_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exstoren")


/*******************************************************************************
 *
```



```
 * FUNCTION:    AcpiExResolveObject
 *
 * PARAMETERS:  SourceDescPtr       - Pointer to the source object
 *              TargetType          - Current type of the target
 *              WalkState           - Current walk state
 *
 * RETURN:      Status, resolved object in SourceDescPtr.
 *
 * DESCRIPTION: Resolve an object. If the object is a reference, dereference
 *              it and return the actual object in the SourceDescPtr.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiExResolveObject (
    ACPI_OPERAND_OBJECT     **SourceDescPtr,
    ACPI_OBJECT_TYPE        TargetType,
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_OPERAND_OBJECT     *SourceDesc = *SourceDescPtr;
    ACPI_STATUS             Status = AE_OK;


    ACPI_FUNCTION_TRACE (ExResolveObject);


    /* Ensure we have a Target that can be stored to */

    switch (TargetType)
    {
    case ACPI_TYPE_BUFFER_FIELD:
    case ACPI_TYPE_LOCAL_REGION_FIELD:
    case ACPI_TYPE_LOCAL_BANK_FIELD:
    case ACPI_TYPE_LOCAL_INDEX_FIELD:
        /*
         * These cases all require only Integers or values that
         * can be converted to Integers (Strings or Buffers)
         */

    case ACPI_TYPE_INTEGER:
    case ACPI_TYPE_STRING:
    case ACPI_TYPE_BUFFER:

        /*
         * Stores into a Field/Region or into a Integer/Buffer/String
         * are all essentially the same. This case handles the
         * "interchangeable" types Integer, String, and Buffer.
         */
        if (SourceDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE)
        {
            /* Resolve a reference object first */

            Status = AcpiExResolveToValue (SourceDescPtr, WalkState);
            if (ACPI_FAILURE (Status))
            {
                break;
            }
        }

        /* For CopyObject, no further validation necessary */

        if (WalkState->Opcode == AML_COPY_OP)
        {
            break;
        }
```

```
123         /* Must have a Integer, Buffer, or String */

125         if ((SourceDesc->Common.Type != ACPI_TYPE_INTEGER)    &&
126             (SourceDesc->Common.Type != ACPI_TYPE_BUFFER)     &&
127             (SourceDesc->Common.Type != ACPI_TYPE_STRING)     &&
128             !((SourceDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE) &&
129                     (SourceDesc->Reference.Class== ACPI_REFCLASS_TABLE)))
130         {
131             /* Conversion successful but still not a valid type */

133             ACPI_ERROR ((AE_INFO,
134                 "Cannot assign type %s to %s (must be type Int/Str/Buf)",
135                 AcpiUtGetObjectTypeName (SourceDesc),
136                 AcpiUtGetTypeName (TargetType)));
137             Status = AE_AML_OPERAND_TYPE;
138         }
139         break;


141     case ACPI_TYPE_LOCAL_ALIAS:
142     case ACPI_TYPE_LOCAL_METHOD_ALIAS:

143         /*
144          * All aliases should have been resolved earlier, during the
145          * operand resolution phase.
146          */
147         ACPI_ERROR ((AE_INFO, "Store into an unresolved Alias object"));
148         Status = AE_AML_INTERNAL;
149         break;


151     case ACPI_TYPE_PACKAGE:
152     default:

153         /*
154          * All other types than Alias and the various Fields come here,
155          * including the untyped case - ACPI_TYPE_ANY.
156          */
157         break;
158     }

160     return_ACPI_STATUS (Status);
161 }


164 /*******************************************************************************
165  *
166  * FUNCTION:    AcpiExStoreObjectToObject
167  *
168  * PARAMETERS:  SourceDesc          - Object to store
169  *              DestDesc            - Object to receive a copy of the source
170  *              NewDesc             - New object if DestDesc is obsoleted
171  *              WalkState           - Current walk state
172  *
173  * RETURN:      Status
174  *
175  * DESCRIPTION: "Store" an object to another object. This may include
176  *              converting the source type to the target type (implicit
177  *              conversion), and a copy of the value of the source to
178  *              the target.
179  *
180  *              The Assignment of an object to another (not named) object
181  *              is handled here.
182  *              The Source passed in will replace the current value (if any)
183  *              with the input value.
184  *
```



```
185  *              When storing into an object the data is converted to the
186  *              target object type then stored in the object. This means
187  *              that the target object type (for an initialized target) will
188  *              not be changed by a store operation.
189  *
190  *              This module allows destination types of Number, String,
191  *              Buffer, and Package.
192  *
193  *              Assumes parameters are already validated. NOTE: SourceDesc
194  *              resolution (from a reference object) must be performed by
195  *              the caller if necessary.
196  *
197  ******************************************************************************/

199 ACPI_STATUS
200 AcpiExStoreObjectToObject (
201     ACPI_OPERAND_OBJECT     *SourceDesc,
202     ACPI_OPERAND_OBJECT     *DestDesc,
203     ACPI_OPERAND_OBJECT     **NewDesc,
204     ACPI_WALK_STATE         *WalkState)
205 {
206     ACPI_OPERAND_OBJECT     *ActualSrcDesc;
207     ACPI_STATUS             Status = AE_OK;


210     ACPI_FUNCTION_TRACE_PTR (ExStoreObjectToObject, SourceDesc);


213     ActualSrcDesc = SourceDesc;
214     if (!DestDesc)
215     {
216         /*
217          * There is no destination object (An uninitialized node or
218          * package element), so we can simply copy the source object
219          * creating a new destination object
220          */
221         Status = AcpiUtCopyIobjectToIobject (ActualSrcDesc, NewDesc, WalkState);
222         return_ACPI_STATUS (Status);
223     }

225     if (SourceDesc->Common.Type != DestDesc->Common.Type)
226     {
227         /*
228          * The source type does not match the type of the destination.
229          * Perform the "implicit conversion" of the source to the current type
230          * of the target as per the ACPI specification.
231          *
232          * If no conversion performed, ActualSrcDesc = SourceDesc.
233          * Otherwise, ActualSrcDesc is a temporary object to hold the
234          * converted object.
235          */
236         Status = AcpiExConvertToTargetType (DestDesc->Common.Type,
237                         SourceDesc, &ActualSrcDesc, WalkState);
238         if (ACPI_FAILURE (Status))
239         {
240             return_ACPI_STATUS (Status);
241         }

243         if (SourceDesc == ActualSrcDesc)
244         {
245             /*
246              * No conversion was performed. Return the SourceDesc as the
247              * new object.
248              */
249             *NewDesc = SourceDesc;
250             return_ACPI_STATUS (AE_OK);
```

```
251         }
252     }

254     /*
255      * We now have two objects of identical types, and we can perform a
256      * copy of the *value* of the source object.
257      */
258     switch (DestDesc->Common.Type)
259     {
260     case ACPI_TYPE_INTEGER:

262         DestDesc->Integer.Value = ActualSrcDesc->Integer.Value;

264         /* Truncate value if we are executing from a 32-bit ACPI table */

266         (void) AcpiExTruncateFor32bitTable (DestDesc);
273         AcpiExTruncateFor32bitTable (DestDesc);
267         break;

269     case ACPI_TYPE_STRING:

271         Status = AcpiExStoreStringToString (ActualSrcDesc, DestDesc);
272         break;

274     case ACPI_TYPE_BUFFER:

276         Status = AcpiExStoreBufferToBuffer (ActualSrcDesc, DestDesc);
277         break;

279     case ACPI_TYPE_PACKAGE:

281         Status = AcpiUtCopyIobjectToIobject (ActualSrcDesc, &DestDesc,
282                     WalkState);
283         break;

285     default:
286         /*
287          * All other types come here.
288          */
289         ACPI_WARNING ((AE_INFO, "Store into type %s not implemented",
290             AcpiUtGetObjectTypeName (DestDesc)));

292         Status = AE_NOT_IMPLEMENTED;
293         break;
294     }

296     if (ActualSrcDesc != SourceDesc)
297     {
298         /* Delete the intermediate (temporary) source object */

300         AcpiUtRemoveReference (ActualSrcDesc);
301     }

303     *NewDesc = DestDesc;
304     return_ACPI_STATUS (Status);
305 }
```

****
7845 Thu Dec 26 13:49:10 2013
new/usr/src/common/acpica/components/executer/exstorob.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exstorob - AML Interpreter object store support, store to object
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXSTOROB_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exstorob")


/******************************************************************************
 *
 * FUNCTION:    AcpiExStoreBufferToBuffer
 *
```



```
 * PARAMETERS:  SourceDesc          - Source object to copy
 *              TargetDesc          - Destination object of the copy
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Copy a buffer object to another buffer object.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiExStoreBufferToBuffer (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc)
{
    UINT32                  Length;
    UINT8                   *Buffer;


    ACPI_FUNCTION_TRACE_PTR (ExStoreBufferToBuffer, SourceDesc);


    /* If Source and Target are the same, just return */

    if (SourceDesc == TargetDesc)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /* We know that SourceDesc is a buffer by now */

    Buffer = ACPI_CAST_PTR (UINT8, SourceDesc->Buffer.Pointer);
    Length = SourceDesc->Buffer.Length;

    /*
     * If target is a buffer of length zero or is a static buffer,
     * allocate a new buffer of the proper length
     */
    if ((TargetDesc->Buffer.Length == 0) ||
        (TargetDesc->Common.Flags & AOPOBJ_STATIC_POINTER))
    {
        TargetDesc->Buffer.Pointer = ACPI_ALLOCATE (Length);
        if (!TargetDesc->Buffer.Pointer)
        {
            return_ACPI_STATUS (AE_NO_MEMORY);
        }

        TargetDesc->Buffer.Length = Length;
    }

    /* Copy source buffer to target buffer */

    if (Length <= TargetDesc->Buffer.Length)
    {
        /* Clear existing buffer and copy in the new one */

        ACPI_MEMSET (TargetDesc->Buffer.Pointer, 0, TargetDesc->Buffer.Length);
        ACPI_MEMCPY (TargetDesc->Buffer.Pointer, Buffer, Length);

#ifdef ACPI_OBSOLETE_BEHAVIOR
        /*
         * NOTE: ACPI versions up to 3.0 specified that the buffer must be
         * truncated if the string is smaller than the buffer. However, "other"
         * implementations of ACPI never did this and thus became the defacto
         * standard. ACPI 3.0A changes this behavior such that the buffer
         * is no longer truncated.
         */
```

```
126         /*
127          * OBSOLETE BEHAVIOR:
128          * If the original source was a string, we must truncate the buffer,
129          * according to the ACPI spec. Integer-to-Buffer and Buffer-to-Buffer
130          * copy must not truncate the original buffer.
131          */
132         if (OriginalSrcType == ACPI_TYPE_STRING)
133         {
134             /* Set the new length of the target */

136             TargetDesc->Buffer.Length = Length;
137         }
138 #endif
139     }
140     else
141     {
142         /* Truncate the source, copy only what will fit */

144         ACPI_MEMCPY (TargetDesc->Buffer.Pointer, Buffer,
145             TargetDesc->Buffer.Length);

147         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
148             "Truncating source buffer from %X to %X\n",
149             Length, TargetDesc->Buffer.Length));
150     }

152     /* Copy flags */

154     TargetDesc->Buffer.Flags = SourceDesc->Buffer.Flags;
155     TargetDesc->Common.Flags &= ~AOPOBJ_STATIC_POINTER;
156     return_ACPI_STATUS (AE_OK);
157 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
    9896 Thu Dec 26 13:49:11 2013
new/usr/src/common/acpica/components/executer/exsystem.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exsystem - Interface to OS services
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXSYSTEM_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exsystem")


/******************************************************************************
 *
 * FUNCTION:    AcpiExSystemWaitSemaphore
 *
 * PARAMETERS:  Semaphore       - Semaphore to wait on
```



```
 *              Timeout         - Max time to wait
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Implements a semaphore wait with a check to see if the
 *              semaphore is available immediately. If it is not, the
 *              interpreter is released before waiting.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiExSystemWaitSemaphore (
    ACPI_SEMAPHORE          Semaphore,
    UINT16                  Timeout)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (ExSystemWaitSemaphore);


    Status = AcpiOsWaitSemaphore (Semaphore, 1, ACPI_DO_NOT_WAIT);
    if (ACPI_SUCCESS (Status))
    {
        return_ACPI_STATUS (Status);
    }

    if (Status == AE_TIME)
    {
        /* We must wait, so unlock the interpreter */

        AcpiExRelinquishInterpreter ();

        Status = AcpiOsWaitSemaphore (Semaphore, 1, Timeout);

        ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
            "*** Thread awake after blocking, %s\n",
            AcpiFormatException (Status)));

        /* Reacquire the interpreter */

       AcpiExReacquireInterpreter ();
    }

    return_ACPI_STATUS (Status);
}
```

unchanged_portion_omitted_

****
15228 Thu Dec 26 13:49:11 2013
new/usr/src/common/acpica/components/executer/exutils.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: exutils - interpreter/scanner utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXUTILS_C__

/*
 * DEFINE_AML_GLOBALS is tested in amlcode.h
 * to determine whether certain global names should be "defined" or only
 * "declared" in the current compilation. This enhances maintainability
 * by enabling a single header file to embody all knowledge of the names
 * in question.
 *
 * Exactly one module of any executable should #define DEFINE_GLOBALS
 * before #including the header files which use this convention. The
 * names in question will be defined and initialized in that module,
 * and declared as extern in all other modules which #include those
 * header files.
 */
```



```
#define DEFINE_AML_GLOBALS

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_EXECUTER
        ACPI_MODULE_NAME    ("exutils")

/* Local prototypes */

static UINT32
AcpiExDigitsNeeded (
    UINT64                  Value,
    UINT32                  Base);


#ifndef ACPI_NO_METHOD_EXECUTION
/*******************************************************************************
 *
 * FUNCTION:    AcpiExEnterInterpreter
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Enter the interpreter execution region. Failure to enter
 *              the interpreter region is a fatal system error. Used in
 *              conjunction with ExitInterpreter.
 *
 ******************************************************************************/

void
AcpiExEnterInterpreter (
    void)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (ExEnterInterpreter);


    Status = AcpiUtAcquireMutex (ACPI_MTX_INTERPRETER);
    if (ACPI_FAILURE (Status))
    {
        ACPI_ERROR ((AE_INFO, "Could not acquire AML Interpreter mutex"));
    }

    return_VOID;
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiExReacquireInterpreter
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Reacquire the interpreter execution region from within the
 *              interpreter code. Failure to enter the interpreter region is a
 *              fatal system error. Used in conjunction with
 *              fatal system error. Used in  conjuction with
```

```
 *              RelinquishInterpreter
 *
 *****************************************************************************/

void
AcpiExReacquireInterpreter (
    void)
{
    ACPI_FUNCTION_TRACE (ExReacquireInterpreter);


    /*
     * If the global serialized flag is set, do not release the interpreter,
     * since it was not actually released by AcpiExRelinquishInterpreter.
     * This forces the interpreter to be single threaded.
     */
    if (!AcpiGbl_AllMethodsSerialized)
    {
        AcpiExEnterInterpreter ();
    }

    return_VOID;
}
```

unchanged_portion_omitted_

```
/*****************************************************************************
 *
 * FUNCTION:    AcpiExTruncateFor32bitTable
 *
 * PARAMETERS:  ObjDesc         - Object to be truncated
 *
 * RETURN:      TRUE if a truncation was performed, FALSE otherwise.
 * RETURN:      none
 *
 * DESCRIPTION: Truncate an ACPI Integer to 32 bits if the execution mode is
 *              32-bit, as determined by the revision of the DSDT.
 *
 *****************************************************************************/

BOOLEAN
void
AcpiExTruncateFor32bitTable (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{

    ACPI_FUNCTION_ENTRY ();


    /*
     * Object must be a valid number and we must be executing
     * a control method. Object could be NS node for AML_INT_NAMEPATH_OP.
     * a control method. NS node could be there for AML_INT_NAMEPATH_OP.
     */
    if ((!ObjDesc) ||
        (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND) ||
        (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
    {
        return (FALSE);
        return;
    }

    if ((AcpiGbl_IntegerByteWidth == 4) &&
        (ObjDesc->Integer.Value > (UINT64) ACPI_UINT32_MAX))
    if (AcpiGbl_IntegerByteWidth == 4)
    {
```



```
        /*
         * We are executing in a 32-bit ACPI table.
         * We are running a method that exists in a 32-bit ACPI table.
         * Truncate the value to 32 bits by zeroing out the upper 32-bit field
         */
        ObjDesc->Integer.Value &= (UINT64) ACPI_UINT32_MAX;
        return (TRUE);
    }

    return (FALSE);
}
```

unchanged_portion_omitted_

```
/*****************************************************************************
 *
 * FUNCTION:    AcpiIsValidSpaceId
 *
 * PARAMETERS:  SpaceId             - ID to be validated
 *
 * RETURN:      TRUE if valid/supported ID.
 *
 * DESCRIPTION: Validate an operation region SpaceID.
 *
 *****************************************************************************/

BOOLEAN
AcpiIsValidSpaceId (
    UINT8                   SpaceId)
{

    if ((SpaceId >= ACPI_NUM_PREDEFINED_REGIONS) &&
        (SpaceId < ACPI_USER_REGION_BEGIN) &&
        (SpaceId != ACPI_ADR_SPACE_DATA_TABLE) &&
        (SpaceId != ACPI_ADR_SPACE_FIXED_HARDWARE))
    {
        return (FALSE);
    }

    return (TRUE);
}


#endif
```

```
**********************************************************
    6700 Thu Dec 26 13:49:12 2013
new/usr/src/common/acpica/components/hardware/hwacpi.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: hwacpi - ACPI Hardware Initialization/Mode Interface
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __HWACPI_C__

#include "acpi.h"
#include "accommon.h"


#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwacpi")


#if (!ACPI_REDUCED_HARDWARE) /* Entire module */
/******************************************************************************
 *
 * FUNCTION:    AcpiHwSetMode
 *
```



```
 * PARAMETERS:  Mode            - SYS_MODE_ACPI or SYS_MODE_LEGACY
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Transitions the system into the requested mode.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiHwSetMode (
    UINT32                  Mode)
{

    ACPI_STATUS             Status;
    UINT32                  Retry;


    ACPI_FUNCTION_TRACE (HwSetMode);


    /* If the Hardware Reduced flag is set, machine is always in acpi mode */

    if (AcpiGbl_ReducedHardware)
    {
        return_ACPI_STATUS (AE_OK);
    }

    /*
     * ACPI 2.0 clarified that if SMI_CMD in FADT is zero,
     * system does not support mode transition.
     */
    if (!AcpiGbl_FADT.SmiCommand)
    {
        ACPI_ERROR ((AE_INFO, "No SMI_CMD in FADT, mode transition failed"));
        return_ACPI_STATUS (AE_NO_HARDWARE_RESPONSE);
    }

    /*
     * ACPI 2.0 clarified the meaning of ACPI_ENABLE and ACPI_DISABLE
     * in FADT: If it is zero, enabling or disabling is not supported.
     * As old systems may have used zero for mode transition,
     * we make sure both the numbers are zero to determine these
     * transitions are not supported.
     */
    if (!AcpiGbl_FADT.AcpiEnable && !AcpiGbl_FADT.AcpiDisable)
    {
        ACPI_ERROR ((AE_INFO,
            "No ACPI mode transition supported in this system "
            "(enable/disable both zero)"));
        return_ACPI_STATUS (AE_OK);
    }

    switch (Mode)
    {
    case ACPI_SYS_MODE_ACPI:

        /* BIOS should have disabled ALL fixed and GP events */

        Status = AcpiHwWritePort (AcpiGbl_FADT.SmiCommand,
                    (UINT32) AcpiGbl_FADT.AcpiEnable, 8);
        ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Attempting to enable ACPI mode\n"));
        break;

    case ACPI_SYS_MODE_LEGACY:

        /*
```

```
124          * BIOS should clear all fixed status bits and restore fixed event
125          * enable bits to default
126          */
127         Status = AcpiHwWritePort (AcpiGbl_FADT.SmiCommand,
128                     (UINT32) AcpiGbl_FADT.AcpiDisable, 8);
129         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
130                     "Attempting to enable Legacy (non-ACPI) mode\n"));
131         break;

133     default:

135         return_ACPI_STATUS (AE_BAD_PARAMETER);
136     }

138     if (ACPI_FAILURE (Status))
139     {
140         ACPI_EXCEPTION ((AE_INFO, Status,
141             "Could not write ACPI mode change"));
142         return_ACPI_STATUS (Status);
143     }

145     /*
146      * Some hardware takes a LONG time to switch modes. Give them 3 sec to
147      * do so, but allow faster systems to proceed more quickly.
148      */
149     Retry = 3000;
150     while (Retry)
151     {
152         if (AcpiHwGetMode () == Mode)
144         if (AcpiHwGetMode() == Mode)
153         {
154             ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Mode %X successfully enabled\n",
155                 Mode));
156             return_ACPI_STATUS (AE_OK);
157         }
158         AcpiOsStall (ACPI_USEC_PER_MSEC);
150         AcpiOsStall(1000);
159         Retry--;
160     }

162     ACPI_ERROR ((AE_INFO, "Hardware did not change modes"));
163     return_ACPI_STATUS (AE_NO_HARDWARE_RESPONSE);
164 }


167 /******************************************************************************
168  *
169  * FUNCTION:    AcpiHwGetMode
170  *
171  * PARAMETERS:  none
172  *
173  * RETURN:      SYS_MODE_ACPI or SYS_MODE_LEGACY
174  *
175  * DESCRIPTION: Return current operating state of system. Determined by
176  *              querying the SCI_EN bit.
177  *
178  ******************************************************************************/

180 UINT32
181 AcpiHwGetMode (
182     void)
183 {
184     ACPI_STATUS             Status;
185     UINT32                  Value;
```



```
188     ACPI_FUNCTION_TRACE (HwGetMode);


191     /* If the Hardware Reduced flag is set, machine is always in acpi mode */

193     if (AcpiGbl_ReducedHardware)
194     {
195         return_UINT32 (ACPI_SYS_MODE_ACPI);
196     }

198     /*
199      * ACPI 2.0 clarified that if SMI_CMD in FADT is zero,
200      * system does not support mode transition.
201      */
202     if (!AcpiGbl_FADT.SmiCommand)
203     {
204         return_UINT32 (ACPI_SYS_MODE_ACPI);
205     }

207     Status = AcpiReadBitRegister (ACPI_BITREG_SCI_ENABLE, &Value);
208     if (ACPI_FAILURE (Status))
209     {
210         return_UINT32 (ACPI_SYS_MODE_LEGACY);
211     }

213     if (Value)
214     {
215         return_UINT32 (ACPI_SYS_MODE_ACPI);
216     }
217     else
218     {
219         return_UINT32 (ACPI_SYS_MODE_LEGACY);
220     }
221 }

223 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
    8029 Thu Dec 26 13:49:12 2013
new/usr/src/common/acpica/components/hardware/hwesleep.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Name: hwesleep.c - ACPI Hardware Sleep/Wake Support functions for the
 *                    extended FADT-V5 sleep registers.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwesleep")


/******************************************************************************
 *
 * FUNCTION:    AcpiHwExecuteSleepMethod
 *
 * PARAMETERS:  MethodPathname      - Pathname of method to execute
 *              IntegerArgument     - Argument to pass to the method
 *
 * RETURN:      None
 *
 * DESCRIPTION: Execute a sleep/wake related method with one integer argument
```



```
 *              and no return value.
 *
 *****************************************************************************/

void
AcpiHwExecuteSleepMethod (
    char                    *MethodPathname,
    UINT32                  IntegerArgument)
{
    ACPI_OBJECT_LIST        ArgList;
    ACPI_OBJECT             Arg;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (HwExecuteSleepMethod);


    /* One argument, IntegerArgument; No return value expected */

    ArgList.Count = 1;
    ArgList.Pointer = &Arg;
    Arg.Type = ACPI_TYPE_INTEGER;
    Arg.Integer.Value = (UINT64) IntegerArgument;

    Status = AcpiEvaluateObject (NULL, MethodPathname, &ArgList, NULL);
    if (ACPI_FAILURE (Status) && Status != AE_NOT_FOUND)
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "While executing method %s",
            MethodPathname));
    }

    return_VOID;
}


/******************************************************************************
 *
 * FUNCTION:    AcpiHwExtendedSleep
 *
 * PARAMETERS:  SleepState          - Which sleep state to enter
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Enter a system sleep state via the extended FADT sleep
 *              registers (V5 FADT).
 *              THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
 *
 *****************************************************************************/

ACPI_STATUS
AcpiHwExtendedSleep (
    UINT8                   SleepState)
{
    ACPI_STATUS             Status;
    UINT8                   SleepTypeValue;
    UINT64                  SleepStatus;


    ACPI_FUNCTION_TRACE (HwExtendedSleep);


    /* Extended sleep registers must be valid */

    if (!AcpiGbl_FADT.SleepControl.Address ||
        !AcpiGbl_FADT.SleepStatus.Address)
    {
```

```
128         return_ACPI_STATUS (AE_NOT_EXIST);
129     }

131     /* Clear wake status (WAK_STS) */

133     Status = AcpiWrite ((UINT64) ACPI_X_WAKE_STATUS, &AcpiGbl_FADT.SleepStatus);
134     if (ACPI_FAILURE (Status))
135     {
136         return_ACPI_STATUS (Status);
137     }

139     AcpiGbl_SystemAwakeAndRunning = FALSE;

141     /* Flush caches, as per ACPI specification */

143     ACPI_FLUSH_CPU_CACHE ();

145     /*
146      * Set the SLP_TYP and SLP_EN bits.
147      *
148      * Note: We only use the first value returned by the \_Sx method
149      * (AcpiGbl_SleepTypeA) - As per ACPI specification.
150      */
151     ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
152         "Entering sleep state [S%u]\n", SleepState));

154     SleepTypeValue = ((AcpiGbl_SleepTypeA << ACPI_X_SLEEP_TYPE_POSITION) &
155         ACPI_X_SLEEP_TYPE_MASK);

157     Status = AcpiWrite ((UINT64) (SleepTypeValue | ACPI_X_SLEEP_ENABLE),
158         &AcpiGbl_FADT.SleepControl);
159     if (ACPI_FAILURE (Status))
160     {
161         return_ACPI_STATUS (Status);
162     }

164     /* Wait for transition back to Working State */

166     do
167     {
168         Status = AcpiRead (&SleepStatus, &AcpiGbl_FADT.SleepStatus);
169         if (ACPI_FAILURE (Status))
170         {
171             return_ACPI_STATUS (Status);
172         }

174     } while (!(((UINT8) SleepStatus) & ACPI_X_WAKE_STATUS));

176     return_ACPI_STATUS (AE_OK);
177 }


180 /*******************************************************************************
181  *
182  * FUNCTION:    AcpiHwExtendedWakePrep
183  *
184  * PARAMETERS:  SleepState          - Which sleep state we just exited
185  *
186  * RETURN:      Status
187  *
188  * DESCRIPTION: Perform first part of OS-independent ACPI cleanup after
189  *              a sleep. Called with interrupts ENABLED.
190  *
191  ******************************************************************************/

193 ACPI_STATUS
```



```
194 AcpiHwExtendedWakePrep (
195     UINT8                   SleepState)
196 {
197     ACPI_STATUS             Status;
198     UINT8                   SleepTypeValue;


201     ACPI_FUNCTION_TRACE (HwExtendedWakePrep);


204     Status = AcpiGetSleepTypeData (ACPI_STATE_S0,
205                 &AcpiGbl_SleepTypeA, &AcpiGbl_SleepTypeB);
206     if (ACPI_SUCCESS (Status))
207     {
208         SleepTypeValue = ((AcpiGbl_SleepTypeA << ACPI_X_SLEEP_TYPE_POSITION) &
209             ACPI_X_SLEEP_TYPE_MASK);

211         (void) AcpiWrite ((UINT64) (SleepTypeValue | ACPI_X_SLEEP_ENABLE),
212             &AcpiGbl_FADT.SleepControl);
213     }

215     return_ACPI_STATUS (AE_OK);
216 }


219 /*******************************************************************************
220  *
221  * FUNCTION:    AcpiHwExtendedWake
222  *
223  * PARAMETERS:  SleepState          - Which sleep state we just exited
224  *
225  * RETURN:      Status
226  *
227  * DESCRIPTION: Perform OS-independent ACPI cleanup after a sleep
228  *              Called with interrupts ENABLED.
229  *
230  ******************************************************************************/

232 ACPI_STATUS
233 AcpiHwExtendedWake (
234     UINT8                   SleepState)
235 {
236     ACPI_FUNCTION_TRACE (HwExtendedWake);


239     /* Ensure EnterSleepStatePrep -> EnterSleepState ordering */

241     AcpiGbl_SleepTypeA = ACPI_SLEEP_TYPE_INVALID;

243     /* Execute the wake methods */

245     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WAKING);
246     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__WAK, SleepState);

248     /*
249      * Some BIOS code assumes that WAK_STS will be cleared on resume
250      * and use it to determine whether the system is rebooting or
251      * resuming. Clear WAK_STS for compatibility.
252      */
253     (void) AcpiWrite ((UINT64) ACPI_X_WAKE_STATUS, &AcpiGbl_FADT.SleepStatus);
254     AcpiGbl_SystemAwakeAndRunning = TRUE;

256     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WORKING);
257     return_ACPI_STATUS (AE_OK);
258 }
```

****
14266 Thu Dec 26 13:49:12 2013
new/usr/src/common/acpica/components/hardware/hwgpe.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: hwgpe - Low level GPE enable/disable/clear functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwgpe")

#if (!ACPI_REDUCED_HARDWARE) /* Entire module */

/* Local prototypes */

static ACPI_STATUS
AcpiHwEnableWakeupGpeBlock (
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
```



```
    void                    *Context);


/******************************************************************************
 *
 * FUNCTION:    AcpiHwGetGpeRegisterBit
 *
 * PARAMETERS:  GpeEventInfo        - Info block for the GPE
 *              GpeRegisterInfo     - Info block for the GPE register
 *
 * RETURN:      Register mask with a one in the GPE bit position
 *
 * DESCRIPTION: Compute the register mask for this GPE. One bit is set in the
 *              correct position for the input GPE.
 *
 ******************************************************************************/

UINT32
AcpiHwGetGpeRegisterBit (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo)
    ACPI_GPE_EVENT_INFO     *GpeEventInfo,
    ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo)
{

    return ((UINT32) 1 <<
        (GpeEventInfo->GpeNumber - GpeEventInfo->RegisterInfo->BaseGpeNumber));
        (GpeEventInfo->GpeNumber - GpeRegisterInfo->BaseGpeNumber));
}


/******************************************************************************
 *
 * FUNCTION:    AcpiHwLowSetGpe
 *
 * PARAMETERS:  GpeEventInfo        - Info block for the GPE to be disabled
 *              Action              - Enable or disable
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Enable or disable a single GPE in the parent enable register.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiHwLowSetGpe (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo,
    UINT32                  Action)
{
    ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
    ACPI_STATUS             Status;
    UINT32                  EnableMask;
    UINT32                  RegisterBit;


    ACPI_FUNCTION_ENTRY ();


    /* Get the info block for the entire GPE register */

    GpeRegisterInfo = GpeEventInfo->RegisterInfo;
    if (!GpeRegisterInfo)
    {
        return (AE_NOT_EXIST);
    }

    /* Get current value of the enable register that contains this GPE */
```

```
122     Status = AcpiHwRead (&EnableMask, &GpeRegisterInfo->EnableAddress);
123     if (ACPI_FAILURE (Status))
124     {
125         return (Status);
126     }

128     /* Set or clear just the bit that corresponds to this GPE */

130     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo);
131     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);
131     switch (Action)
132     {
133     case ACPI_GPE_CONDITIONAL_ENABLE:

135         /* Only enable if the EnableForRun bit is set */

137         if (!(RegisterBit & GpeRegisterInfo->EnableForRun))
138         {
139             return (AE_BAD_PARAMETER);
140         }

142         /*lint -fallthrough */

144     case ACPI_GPE_ENABLE:

146         ACPI_SET_BIT (EnableMask, RegisterBit);
147         break;

149     case ACPI_GPE_DISABLE:

151         ACPI_CLEAR_BIT (EnableMask, RegisterBit);
152         break;

154     default:

156         ACPI_ERROR ((AE_INFO, "Invalid GPE Action, %u", Action));
154         ACPI_ERROR ((AE_INFO, "Invalid GPE Action, %u\n", Action));
157         return (AE_BAD_PARAMETER);
158     }

160     /* Write the updated enable mask */

162     Status = AcpiHwWrite (EnableMask, &GpeRegisterInfo->EnableAddress);
163     return (Status);
164 }


167 /******************************************************************************
168  *
169  * FUNCTION:    AcpiHwClearGpe
170  *
171  * PARAMETERS:  GpeEventInfo        - Info block for the GPE to be cleared
172  *
173  * RETURN:      Status
174  *
175  * DESCRIPTION: Clear the status bit for a single GPE.
176  *
177  *****************************************************************************/

179 ACPI_STATUS
180 AcpiHwClearGpe (
181     ACPI_GPE_EVENT_INFO     *GpeEventInfo)
182 {
183     ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
184     ACPI_STATUS             Status;
```



```
185     UINT32                  RegisterBit;


188     ACPI_FUNCTION_ENTRY ();

190     /* Get the info block for the entire GPE register */

192     GpeRegisterInfo = GpeEventInfo->RegisterInfo;
193     if (!GpeRegisterInfo)
194     {
195         return (AE_NOT_EXIST);
196     }

198     /*
199      * Write a one to the appropriate bit in the status register to
200      * clear this GPE.
201      */
202     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo);
200     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);

204     Status = AcpiHwWrite (RegisterBit,
205                     &GpeRegisterInfo->StatusAddress);

207     return (Status);
208 }


211 /******************************************************************************
212  *
213  * FUNCTION:    AcpiHwGetGpeStatus
214  *
215  * PARAMETERS:  GpeEventInfo        - Info block for the GPE to queried
216  *              EventStatus         - Where the GPE status is returned
217  *
218  * RETURN:      Status
219  *
220  * DESCRIPTION: Return the status of a single GPE.
221  *
222  *****************************************************************************/

224 ACPI_STATUS
225 AcpiHwGetGpeStatus (
226     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
227     ACPI_EVENT_STATUS       *EventStatus)
228 {
229     UINT32                  InByte;
230     UINT32                  RegisterBit;
231     ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo;
232     ACPI_EVENT_STATUS       LocalEventStatus = 0;
233     ACPI_STATUS             Status;


236     ACPI_FUNCTION_ENTRY ();


239     if (!EventStatus)
240     {
241         return (AE_BAD_PARAMETER);
242     }

244     /* Get the info block for the entire GPE register */

246     GpeRegisterInfo = GpeEventInfo->RegisterInfo;

248     /* Get the register bitmask for this GPE */
```

```
250     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo);
248     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);

252     /* GPE currently enabled? (enabled for runtime?) */

254     if (RegisterBit & GpeRegisterInfo->EnableForRun)
255     {
256         LocalEventStatus |= ACPI_EVENT_FLAG_ENABLED;
257     }

259     /* GPE enabled for wake? */

261     if (RegisterBit & GpeRegisterInfo->EnableForWake)
262     {
263         LocalEventStatus |= ACPI_EVENT_FLAG_WAKE_ENABLED;
264     }

266     /* GPE currently active (status bit == 1)? */

268     Status = AcpiHwRead (&InByte, &GpeRegisterInfo->StatusAddress);
269     if (ACPI_FAILURE (Status))
270     {
271         return (Status);
272     }

274     if (RegisterBit & InByte)
275     {
276         LocalEventStatus |= ACPI_EVENT_FLAG_SET;
277     }

279     /* Set return value */

281     (*EventStatus) = LocalEventStatus;
282     return (AE_OK);
283 }
```

*unchanged_portion_omitted_*

```
543 #endif /* !ACPI_REDUCED_HARDWARE */
```

****
14483 Thu Dec 26 13:49:13 2013
new/usr/src/common/acpica/components/hardware/hwpci.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: hwpci - Obtain PCI bus, device, and function numbers
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __HWPCI_C__

#include "acpi.h"
#include "accommon.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("hwpci")


/* PCI configuration space values */

#define PCI_CFG_HEADER_TYPE_REG             0x0E
#define PCI_CFG_PRIMARY_BUS_NUMBER_REG      0x18
#define PCI_CFG_SECONDARY_BUS_NUMBER_REG    0x19
```



```
/* PCI header values */

#define PCI_HEADER_TYPE_MASK                0x7F
#define PCI_TYPE_BRIDGE                     0x01
#define PCI_TYPE_CARDBUS_BRIDGE             0x02

typedef struct acpi_pci_device
{
    ACPI_HANDLE             Device;
    struct acpi_pci_device  *Next;

} ACPI_PCI_DEVICE;
```

_____unchanged_portion_omitted_

```
/******************************************************************************
 *
 * FUNCTION:    AcpiHwProcessPciList
 *
 * PARAMETERS:  PciId               - Initial values for the PCI ID. May be
 *                                    modified by this function.
 *              ListHead            - Device list created by
 *                                    AcpiHwBuildPciList
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk downward through the PCI device list, getting the device
 *              info for each, via the PCI configuration space and updating
 *              the PCI ID as necessary. Deletes the list during traversal.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiHwProcessPciList (
    ACPI_PCI_ID             *PciId,
    ACPI_PCI_DEVICE         *ListHead)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_PCI_DEVICE         *Info;
    UINT16                  BusNumber;
    BOOLEAN                 IsBridge = TRUE;


    ACPI_FUNCTION_NAME (HwProcessPciList);


    ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
        "Input PciId:  Seg %4.4X Bus %4.4X Dev %4.4X Func %4.4X\n",
        PciId->Segment, PciId->Bus, PciId->Device, PciId->Function));

    BusNumber = PciId->Bus;

    /*
     * Descend down the namespace tree, collecting PCI device, function,
     * and bus numbers. BusNumber is only important for PCI bridges.
     * Algorithm: As we descend the tree, use the last valid PCI device,
     * function, and bus numbers that are discovered, and assign them
     * to the PCI ID for the target device.
     */
    Info = ListHead;
    while (Info)
    {
        Status = AcpiHwGetPciDeviceInfo (PciId, Info->Device,
            &BusNumber, &IsBridge);
        if (ACPI_FAILURE (Status))
        {
```

```
295             return (Status);
295             return_ACPI_STATUS (Status);
296         }

298         Info = Info->Next;
299     }

301     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
302         "Output PciId: Seg %4.4X Bus %4.4X Dev %4.4X Func %4.4X "
303         "Status %X BusNumber %X IsBridge %X\n",
304         PciId->Segment, PciId->Bus, PciId->Device, PciId->Function,
305         Status, BusNumber, IsBridge));

307     return (AE_OK);
307     return_ACPI_STATUS (AE_OK);
308 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   21657 Thu Dec 26 13:49:13 2013
new/usr/src/common/acpica/components/hardware/hwregs.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: hwregs - Read/write access functions for the various ACPI
 *                       control and status registers.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __HWREGS_C__

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwregs")


#if (!ACPI_REDUCED_HARDWARE)

/* Local Prototypes */
```



```
static ACPI_STATUS
AcpiHwReadMultiple (
    UINT32                  *Value,
    ACPI_GENERIC_ADDRESS    *RegisterA,
    ACPI_GENERIC_ADDRESS    *RegisterB);

static ACPI_STATUS
AcpiHwWriteMultiple (
    UINT32                  Value,
    ACPI_GENERIC_ADDRESS    *RegisterA,
    ACPI_GENERIC_ADDRESS    *RegisterB);

#endif /* !ACPI_REDUCED_HARDWARE */

/******************************************************************************
 *
 * FUNCTION:    AcpiHwValidateRegister
 *
 * PARAMETERS:  Reg                 - GAS register structure
 *              MaxBitWidth         - Max BitWidth supported (32 or 64)
 *              Address             - Pointer to where the gas->address
 *                                    is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Validate the contents of a GAS register. Checks the GAS
 *              pointer, Address, SpaceId, BitWidth, and BitOffset.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiHwValidateRegister (
    ACPI_GENERIC_ADDRESS    *Reg,
    UINT8                   MaxBitWidth,
    UINT64                  *Address)
{

    /* Must have a valid pointer to a GAS structure */

    if (!Reg)
    {
        return (AE_BAD_PARAMETER);
    }

    /*
     * Copy the target address. This handles possible alignment issues.
     * Address must not be null. A null address also indicates an optional
     * ACPI register that is not supported, so no error message.
     */
    ACPI_MOVE_64_TO_64 (Address, &Reg->Address);
    if (!(*Address))
    {
        return (AE_BAD_ADDRESS);
    }

    /* Validate the SpaceID */

    if ((Reg->SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
        (Reg->SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
    {
        ACPI_ERROR ((AE_INFO,
            "Unsupported address space: 0x%X", Reg->SpaceId));
        return (AE_SUPPORT);
    }

    /* Validate the BitWidth */
```

```
126     if ((Reg->BitWidth != 8) &&
127         (Reg->BitWidth != 16) &&
128         (Reg->BitWidth != 32) &&
129         (Reg->BitWidth != MaxBitWidth))
130     {
131         ACPI_ERROR ((AE_INFO,
132             "Unsupported register bit width: 0x%X", Reg->BitWidth));
133         return (AE_SUPPORT);
134     }

136     /* Validate the BitOffset. Just a warning for now. */

138     if (Reg->BitOffset != 0)
139     {
140         ACPI_WARNING ((AE_INFO,
141             "Unsupported register bit offset: 0x%X", Reg->BitOffset));
142     }

144     return (AE_OK);
145 }


148 /******************************************************************************
149  *
150  * FUNCTION:    AcpiHwRead
151  *
152  * PARAMETERS:  Value               - Where the value is returned
153  *              Reg                 - GAS register structure
154  *
155  * RETURN:      Status
156  *
157  * DESCRIPTION: Read from either memory or IO space. This is a 32-bit max
158  *              version of AcpiRead, used internally since the overhead of
159  *              64-bit values is not needed.
160  *
161  * LIMITATIONS: <These limitations also apply to AcpiHwWrite>
162  *      BitWidth must be exactly 8, 16, or 32.
163  *      SpaceID must be SystemMemory or SystemIO.
164  *      BitOffset and AccessWidth are currently ignored, as there has
165  *          not been a need to implement these.
166  *
167  *****************************************************************************/

169 ACPI_STATUS
170 AcpiHwRead (
171     UINT32                  *Value,
172     ACPI_GENERIC_ADDRESS    *Reg)
173 {
174     UINT64                  Address;
175     UINT64                  Value64;
176     ACPI_STATUS             Status;


179     ACPI_FUNCTION_NAME (HwRead);


182     /* Validate contents of the GAS register */

184     Status = AcpiHwValidateRegister (Reg, 32, &Address);
185     if (ACPI_FAILURE (Status))
186     {
187         return (Status);
188     }

190     /* Initialize entire 32-bit return value to zero */
```



```
192     *Value = 0;

194     /*
195      * Two address spaces supported: Memory or IO. PCI_Config is
196      * not supported here because the GAS structure is insufficient
197      */
198     if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
199     {
200         Status = AcpiOsReadMemory ((ACPI_PHYSICAL_ADDRESS)
201                     Address, &Value64, Reg->BitWidth);

203         *Value = (UINT32) Value64;
198                     Address, Value, Reg->BitWidth);
204     }
205     else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
206     {
207         Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
208                     Address, Value, Reg->BitWidth);
209     }

211     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
212         "Read:  %8.8X width %2d from %8.8X%8.8X (%s)\n",
213         *Value, Reg->BitWidth, ACPI_FORMAT_UINT64 (Address),
214         AcpiUtGetRegionName (Reg->SpaceId)));

216     return (Status);
217 }


220 /******************************************************************************
221  *
222  * FUNCTION:    AcpiHwWrite
223  *
224  * PARAMETERS:  Value               - Value to be written
225  *              Reg                 - GAS register structure
226  *
227  * RETURN:      Status
228  *
229  * DESCRIPTION: Write to either memory or IO space. This is a 32-bit max
230  *              version of AcpiWrite, used internally since the overhead of
231  *              64-bit values is not needed.
232  *
233  *****************************************************************************/

235 ACPI_STATUS
236 AcpiHwWrite (
237     UINT32                  Value,
238     ACPI_GENERIC_ADDRESS    *Reg)
239 {
240     UINT64                  Address;
241     ACPI_STATUS             Status;


244     ACPI_FUNCTION_NAME (HwWrite);


247     /* Validate contents of the GAS register */

249     Status = AcpiHwValidateRegister (Reg, 32, &Address);
250     if (ACPI_FAILURE (Status))
251     {
252         return (Status);
253     }

255     /*
```

```
256      * Two address spaces supported: Memory or IO. PCI_Config is
257      * not supported here because the GAS structure is insufficient
258      */
259     if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
260     {
261         Status = AcpiOsWriteMemory ((ACPI_PHYSICAL_ADDRESS)
262                     Address, (UINT64) Value, Reg->BitWidth);
263                     Address, Value, Reg->BitWidth);
264     }
265     else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
266     {
267         Status = AcpiHwWritePort ((ACPI_IO_ADDRESS)
268                     Address, Value, Reg->BitWidth);
269     }
270
271     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
272         "Wrote: %8.8X width %2d   to %8.8X%8.8X (%s)\n",
273         Value, Reg->BitWidth, ACPI_FORMAT_UINT64 (Address),
274         AcpiUtGetRegionName (Reg->SpaceId)));
275
276     return (Status);
277 }
278
279 #if (!ACPI_REDUCED_HARDWARE)
280 /******************************************************************************
281  *
282  * FUNCTION:    AcpiHwClearAcpiStatus
283  *
284  * PARAMETERS:  None
285  *
286  * RETURN:      Status
287  *
288  * DESCRIPTION: Clears all fixed and general purpose status bits
289  *
290  *****************************************************************************/
291
292 ACPI_STATUS
293 AcpiHwClearAcpiStatus (
294     void)
295 {
296     ACPI_STATUS             Status;
297     ACPI_CPU_FLAGS          LockFlags = 0;
298
299
300     ACPI_FUNCTION_TRACE (HwClearAcpiStatus);
301
302
303     ACPI_DEBUG_PRINT ((ACPI_DB_IO, "About to write %04X to %8.8X%8.8X\n",
304         ACPI_BITMASK_ALL_FIXED_STATUS,
305         ACPI_FORMAT_UINT64 (AcpiGbl_XPm1aStatus.Address)));
306
307     LockFlags = AcpiOsAcquireLock (AcpiGbl_HardwareLock);
308
309     /* Clear the fixed events in PM1 A/B */
310
311     Status = AcpiHwRegisterWrite (ACPI_REGISTER_PM1_STATUS,
312                 ACPI_BITMASK_ALL_FIXED_STATUS);
313     if (ACPI_FAILURE (Status))
314     {
315         goto UnlockAndExit;
316     }
317
318     /* Clear the GPE Bits in all GPE registers in all GPE blocks */
319
320     Status = AcpiEvWalkGpeList (AcpiHwClearGpeBlock, NULL);
```



```
321
322 UnlockAndExit:
323     AcpiOsReleaseLock (AcpiGbl_HardwareLock, LockFlags);
324     return_ACPI_STATUS (Status);
325 }
326
327
328 /******************************************************************************
329  *
330  * FUNCTION:    AcpiHwGetBitRegisterInfo
331  * FUNCTION:    AcpiHwGetRegisterBitMask
332  *
333  * PARAMETERS:  RegisterId          - Index of ACPI Register to access
334  *
335  * RETURN:      The bitmask to be used when accessing the register
336  *
337  * DESCRIPTION: Map RegisterId into a register bitmask.
338  *
339  *****************************************************************************/
340
341 ACPI_BIT_REGISTER_INFO *
342 AcpiHwGetBitRegisterInfo (
343     UINT32                  RegisterId)
344 {
345     ACPI_FUNCTION_ENTRY ();
346
347     if (RegisterId > ACPI_BITREG_MAX)
348     {
349         ACPI_ERROR ((AE_INFO, "Invalid BitRegister ID: 0x%X", RegisterId));
350         return (NULL);
351     }
352
353     return (&AcpiGbl_BitRegisterInfo[RegisterId]);
354 }
_____unchanged_portion_omitted_
399 /******************************************************************************
400  *
401  * FUNCTION:    AcpiHwRegisterRead
402  *
403  * PARAMETERS:  RegisterId          - ACPI Register ID
404  *              ReturnValue         - Where the register value is returned
405  *
406  * RETURN:      Status and the value read.
407  *
408  * DESCRIPTION: Read from the specified ACPI register
409  *
410  *****************************************************************************/
411
412 ACPI_STATUS
413 AcpiHwRegisterRead (
414     UINT32                  RegisterId,
415     UINT32                  *ReturnValue)
416 {
417     UINT32                  Value = 0;
418     ACPI_STATUS             Status;
419
420
421     ACPI_FUNCTION_TRACE (HwRegisterRead);
422
423
424     switch (RegisterId)
425     {
426     case ACPI_REGISTER_PM1_STATUS:          /* PM1 A/B: 16-bit access each */
```

```
428         Status = AcpiHwReadMultiple (&Value,
429                     &AcpiGbl_XPm1aStatus,
430                     &AcpiGbl_XPm1bStatus);
431         break;


433     case ACPI_REGISTER_PM1_ENABLE:          /* PM1 A/B: 16-bit access each */

435         Status = AcpiHwReadMultiple (&Value,
436                     &AcpiGbl_XPm1aEnable,
437                     &AcpiGbl_XPm1bEnable);
438         break;


440     case ACPI_REGISTER_PM1_CONTROL:         /* PM1 A/B: 16-bit access each */

442         Status = AcpiHwReadMultiple (&Value,
443                     &AcpiGbl_FADT.XPm1aControlBlock,
444                     &AcpiGbl_FADT.XPm1bControlBlock);

446         /*
447          * Zero the write-only bits. From the ACPI specification, "Hardware
448          * Write-Only Bits": "Upon reads to registers with write-only bits,
449          * software masks out all write-only bits."
450          */
451         Value &= ~ACPI_PM1_CONTROL_WRITEONLY_BITS;
452         break;


454     case ACPI_REGISTER_PM2_CONTROL:         /* 8-bit access */

456         Status = AcpiHwRead (&Value, &AcpiGbl_FADT.XPm2ControlBlock);
457         break;


459     case ACPI_REGISTER_PM_TIMER:            /* 32-bit access */

461         Status = AcpiHwRead (&Value, &AcpiGbl_FADT.XPmTimerBlock);
462         break;


464     case ACPI_REGISTER_SMI_COMMAND_BLOCK:   /* 8-bit access */

466         Status = AcpiHwReadPort (AcpiGbl_FADT.SmiCommand, &Value, 8);
467         break;

469     default:

469     default:
471         ACPI_ERROR ((AE_INFO, "Unknown Register ID: 0x%X",
472             RegisterId));
473         Status = AE_BAD_PARAMETER;
474         break;
475     }

477     if (ACPI_SUCCESS (Status))
478     {
479         *ReturnValue = Value;
480     }

482     return_ACPI_STATUS (Status);
483 }


486 /*******************************************************************************
```



```
487  *
488  * FUNCTION:    AcpiHwRegisterWrite
489  *
490  * PARAMETERS:  RegisterId          - ACPI Register ID
491  *              Value               - The value to write
492  *
493  * RETURN:      Status
494  *
495  * DESCRIPTION: Write to the specified ACPI register
496  *
497  * NOTE: In accordance with the ACPI specification, this function automatically
498  * preserves the value of the following bits, meaning that these bits cannot be
499  * changed via this interface:
500  *
501  * PM1_CONTROL[0] = SCI_EN
502  * PM1_CONTROL[9]
503  * PM1_STATUS[11]
504  *
505  * ACPI References:
506  * 1) Hardware Ignored Bits: When software writes to a register with ignored
507  *      bit fields, it preserves the ignored bit fields
508  * 2) SCI_EN: OSPM always preserves this bit position
509  *
510  ******************************************************************************/

512 ACPI_STATUS
513 AcpiHwRegisterWrite (
514     UINT32                  RegisterId,
515     UINT32                  Value)
516 {
517     ACPI_STATUS             Status;
518     UINT32                  ReadValue;


521     ACPI_FUNCTION_TRACE (HwRegisterWrite);


524     switch (RegisterId)
525     {
526     case ACPI_REGISTER_PM1_STATUS:          /* PM1 A/B: 16-bit access each */
527         /*
528          * Handle the "ignored" bit in PM1 Status. According to the ACPI
529          * specification, ignored bits are to be preserved when writing.
530          * Normally, this would mean a read/modify/write sequence. However,
531          * preserving a bit in the status register is different. Writing a
532          * one clears the status, and writing a zero preserves the status.
533          * Therefore, we must always write zero to the ignored bit.
534          *
535          * This behavior is clarified in the ACPI 4.0 specification.
536          */
537         Value &= ~ACPI_PM1_STATUS_PRESERVED_BITS;

539         Status = AcpiHwWriteMultiple (Value,
540                     &AcpiGbl_XPm1aStatus,
541                     &AcpiGbl_XPm1bStatus);
542         break;


544     case ACPI_REGISTER_PM1_ENABLE:          /* PM1 A/B: 16-bit access each */

546         Status = AcpiHwWriteMultiple (Value,
547                     &AcpiGbl_XPm1aEnable,
548                     &AcpiGbl_XPm1bEnable);
549         break;
```

```
551     case ACPI_REGISTER_PM1_CONTROL:          /* PM1 A/B: 16-bit access each */

552         /*
553          * Perform a read first to preserve certain bits (per ACPI spec)
554          * Note: This includes SCI_EN, we never want to change this bit
555          */
556         Status = AcpiHwReadMultiple (&ReadValue,
557                     &AcpiGbl_FADT.XPm1aControlBlock,
558                     &AcpiGbl_FADT.XPm1bControlBlock);
559         if (ACPI_FAILURE (Status))
560         {
561             goto Exit;
562         }

564         /* Insert the bits to be preserved */

566         ACPI_INSERT_BITS (Value, ACPI_PM1_CONTROL_PRESERVED_BITS, ReadValue);

568         /* Now we can write the data */

570         Status = AcpiHwWriteMultiple (Value,
571                     &AcpiGbl_FADT.XPm1aControlBlock,
572                     &AcpiGbl_FADT.XPm1bControlBlock);
573         break;


575     case ACPI_REGISTER_PM2_CONTROL:          /* 8-bit access */

576         /*
577          * For control registers, all reserved bits must be preserved,
578          * as per the ACPI spec.
579          */
580         Status = AcpiHwRead (&ReadValue, &AcpiGbl_FADT.XPm2ControlBlock);
581         if (ACPI_FAILURE (Status))
582         {
583             goto Exit;
584         }

586         /* Insert the bits to be preserved */

588         ACPI_INSERT_BITS (Value, ACPI_PM2_CONTROL_PRESERVED_BITS, ReadValue);

590         Status = AcpiHwWrite (Value, &AcpiGbl_FADT.XPm2ControlBlock);
591         break;


593     case ACPI_REGISTER_PM_TIMER:             /* 32-bit access */

595         Status = AcpiHwWrite (Value, &AcpiGbl_FADT.XPmTimerBlock);
596         break;


598     case ACPI_REGISTER_SMI_COMMAND_BLOCK:    /* 8-bit access */

600         /* SMI_CMD is currently always in IO space */

602         Status = AcpiHwWritePort (AcpiGbl_FADT.SmiCommand, Value, 8);
603         break;

605     default:

612     default:
607         ACPI_ERROR ((AE_INFO, "Unknown Register ID: 0x%X",
608             RegisterId));
609         Status = AE_BAD_PARAMETER;
610         break;
```



```
611     }

613 Exit:
614     return_ACPI_STATUS (Status);
615 }
```

*unchanged_portion_omitted_*

```
728 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   10864 Thu Dec 26 13:49:14 2013
new/usr/src/common/acpica/components/hardware/hwsleep.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Name: hwsleep.c - ACPI Hardware Sleep/Wake Support functions for the
 *                   original/legacy sleep/PM registers.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwsleep")


#if (!ACPI_REDUCED_HARDWARE) /* Entire module */
/******************************************************************************
 *
 * FUNCTION:    AcpiHwLegacySleep
 *
 * PARAMETERS:  SleepState          - Which sleep state to enter
 *
 * RETURN:      Status
 *
```



```
 * DESCRIPTION: Enter a system sleep state via the legacy FADT PM registers
 *              THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
 *
 *****************************************************************************/

ACPI_STATUS
AcpiHwLegacySleep (
    UINT8                   SleepState)
{
    ACPI_BIT_REGISTER_INFO  *SleepTypeRegInfo;
    ACPI_BIT_REGISTER_INFO  *SleepEnableRegInfo;
    UINT32                  Pm1aControl;
    UINT32                  Pm1bControl;
    UINT32                  InValue;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (HwLegacySleep);


    SleepTypeRegInfo = AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_TYPE);
    SleepEnableRegInfo = AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_ENABLE);

    /* Clear wake status */

    Status = AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, ACPI_CLEAR_STATUS);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Clear all fixed and general purpose status bits */

    Status = AcpiHwClearAcpiStatus ();
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * 1) Disable/Clear all GPEs
     * 2) Enable all wakeup GPEs
     */
    Status = AcpiHwDisableAllGpes ();
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }
    AcpiGbl_SystemAwakeAndRunning = FALSE;

    Status = AcpiHwEnableAllWakeupGpes ();
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Get current value of PM1A control */

    Status = AcpiHwRegisterRead (ACPI_REGISTER_PM1_CONTROL,
                &Pm1aControl);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }
    ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
        "Entering sleep state [S%u]\n", SleepState));
```

```
128     /* Clear the SLP_EN and SLP_TYP fields */
130     Pm1aControl &= ~(SleepTypeRegInfo->AccessBitMask |
131                      SleepEnableRegInfo->AccessBitMask);
132     Pm1bControl = Pm1aControl;
134     /* Insert the SLP_TYP bits */
136     Pm1aControl |= (AcpiGbl_SleepTypeA << SleepTypeRegInfo->BitPosition);
137     Pm1bControl |= (AcpiGbl_SleepTypeB << SleepTypeRegInfo->BitPosition);
139     /*
140      * We split the writes of SLP_TYP and SLP_EN to workaround
141      * poorly implemented hardware.
142      */
144     /* Write #1: write the SLP_TYP data to the PM1 Control registers */
146     Status = AcpiHwWritePm1Control (Pm1aControl, Pm1bControl);
147     if (ACPI_FAILURE (Status))
148     {
149         return_ACPI_STATUS (Status);
150     }
152     /* Insert the sleep enable (SLP_EN) bit */
154     Pm1aControl |= SleepEnableRegInfo->AccessBitMask;
155     Pm1bControl |= SleepEnableRegInfo->AccessBitMask;
157     /* Flush caches, as per ACPI specification */
159     ACPI_FLUSH_CPU_CACHE ();
161     /* Write #2: Write both SLP_TYP + SLP_EN */
163     Status = AcpiHwWritePm1Control (Pm1aControl, Pm1bControl);
164     if (ACPI_FAILURE (Status))
165     {
166         return_ACPI_STATUS (Status);
167     }
169     if (SleepState > ACPI_STATE_S3)
170     {
171         /*
172          * We wanted to sleep > S3, but it didn't happen (by virtue of the
173          * fact that we are still executing!)
174          *
175          * Wait ten seconds, then try again. This is to get S4/S5 to work on
176          * all machines.
177          *
178          * We wait so long to allow chipsets that poll this reg very slowly
179          * to still read the right value. Ideally, this block would go
180          * away entirely.
181          */
182         AcpiOsStall (10 * ACPI_USEC_PER_SEC);
184         Status = AcpiHwRegisterWrite (ACPI_REGISTER_PM1_CONTROL,
185                     SleepEnableRegInfo->AccessBitMask);
186         if (ACPI_FAILURE (Status))
187         {
188             return_ACPI_STATUS (Status);
189         }
190     }
192     /* Wait for transition back to Working State */
```



```
194     do
195     {
196         Status = AcpiReadBitRegister (ACPI_BITREG_WAKE_STATUS, &InValue);
197         if (ACPI_FAILURE (Status))
198         {
199             return_ACPI_STATUS (Status);
200         }
202     } while (!InValue);
204     return_ACPI_STATUS (AE_OK);
205 }


208 /******************************************************************************
209  *
210  * FUNCTION:    AcpiHwLegacyWakePrep
211  *
212  * PARAMETERS:  SleepState          - Which sleep state we just exited
213  *
214  * RETURN:      Status
215  *
216  * DESCRIPTION: Perform the first state of OS-independent ACPI cleanup after a
217  *              sleep.
218  *              Called with interrupts ENABLED.
219  *
220  *****************************************************************************/
222 ACPI_STATUS
223 AcpiHwLegacyWakePrep (
224     UINT8                   SleepState)
225 {
226     ACPI_STATUS             Status;
227     ACPI_BIT_REGISTER_INFO  *SleepTypeRegInfo;
228     ACPI_BIT_REGISTER_INFO  *SleepEnableRegInfo;
229     UINT32                  Pm1aControl;
230     UINT32                  Pm1bControl;
233     ACPI_FUNCTION_TRACE (HwLegacyWakePrep);
235     /*
236      * Set SLP_TYPE and SLP_EN to state S0.
237      * This is unclear from the ACPI Spec, but it is required
238      * by some machines.
239      */
240     Status = AcpiGetSleepTypeData (ACPI_STATE_S0,
241                 &AcpiGbl_SleepTypeA, &AcpiGbl_SleepTypeB);
242     if (ACPI_SUCCESS (Status))
243     {
244         SleepTypeRegInfo =
245             AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_TYPE);
246         SleepEnableRegInfo =
247             AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_ENABLE);
249         /* Get current value of PM1A control */
251         Status = AcpiHwRegisterRead (ACPI_REGISTER_PM1_CONTROL,
252                     &Pm1aControl);
253         if (ACPI_SUCCESS (Status))
254         {
255             /* Clear the SLP_EN and SLP_TYP fields */
257             Pm1aControl &= ~(SleepTypeRegInfo->AccessBitMask |
258                 SleepEnableRegInfo->AccessBitMask);
```

```
259             Pm1bControl = Pm1aControl;
260
261             /* Insert the SLP_TYP bits */
262
263             Pm1aControl |= (AcpiGbl_SleepTypeA <<
264                 SleepTypeRegInfo->BitPosition);
265             Pm1bControl |= (AcpiGbl_SleepTypeB <<
266                 SleepTypeRegInfo->BitPosition);
267
268             /* Write the control registers and ignore any errors */
269
270             (void) AcpiHwWritePm1Control (Pm1aControl, Pm1bControl);
271         }
272     }
273
274     return_ACPI_STATUS (Status);
275 }
276
277
278 /******************************************************************************
279  *
280  * FUNCTION:    AcpiHwLegacyWake
281  *
282  * PARAMETERS:  SleepState          - Which sleep state we just exited
283  *
284  * RETURN:      Status
285  *
286  * DESCRIPTION: Perform OS-independent ACPI cleanup after a sleep
287  *              Called with interrupts ENABLED.
288  *
289  *****************************************************************************/
290
291 ACPI_STATUS
292 AcpiHwLegacyWake (
293     UINT8                   SleepState)
294 {
295     ACPI_STATUS             Status;
296
297
298     ACPI_FUNCTION_TRACE (HwLegacyWake);
299
300
301     /* Ensure EnterSleepStatePrep -> EnterSleepState ordering */
302
303     AcpiGbl_SleepTypeA = ACPI_SLEEP_TYPE_INVALID;
304     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WAKING);
305
306     /*
307      * GPEs must be enabled before _WAK is called as GPEs
308      * might get fired there
309      *
310      * Restore the GPEs:
311      * 1) Disable/Clear all GPEs
312      * 2) Enable all runtime GPEs
313      */
314     Status = AcpiHwDisableAllGpes ();
315     if (ACPI_FAILURE (Status))
316     {
317         return_ACPI_STATUS (Status);
318     }
319
320     Status = AcpiHwEnableAllRuntimeGpes ();
321     if (ACPI_FAILURE (Status))
322     {
323         return_ACPI_STATUS (Status);
324     }
```



```
325
326     /*
327      * Now we can execute _WAK, etc. Some machines require that the GPEs
328      * are enabled before the wake methods are executed.
329      */
330     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__WAK, SleepState);
331
332     /*
333      * Some BIOS code assumes that WAK_STS will be cleared on resume
334      * and use it to determine whether the system is rebooting or
335      * resuming. Clear WAK_STS for compatibility.
336      */
337     (void) AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, ACPI_CLEAR_STATUS);
338     AcpiGbl_SystemAwakeAndRunning = TRUE;
339
340     /* Enable power button */
341
342     (void) AcpiWriteBitRegister(
343             AcpiGbl_FixedEventInfo[ACPI_EVENT_POWER_BUTTON].EnableRegisterId,
344             ACPI_ENABLE_EVENT);
345
346     (void) AcpiWriteBitRegister(
347             AcpiGbl_FixedEventInfo[ACPI_EVENT_POWER_BUTTON].StatusRegisterId,
348             ACPI_CLEAR_STATUS);
349
350     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WORKING);
351     return_ACPI_STATUS (Status);
352 }
353
354 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
    7124 Thu Dec 26 13:49:14 2013
new/usr/src/common/acpica/components/hardware/hwtimer.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: hwtimer.c - ACPI Power Management Timer Interface
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwtimer")


#if (!ACPI_REDUCED_HARDWARE) /* Entire module */
/******************************************************************************
 *
 * FUNCTION:    AcpiGetTimerResolution
 *
```



```
 * PARAMETERS:  Resolution          - Where the resolution is returned
 *
 * RETURN:      Status and timer resolution
 *
 * DESCRIPTION: Obtains resolution of the ACPI PM Timer (24 or 32 bits).
 *
 *****************************************************************************/

ACPI_STATUS
AcpiGetTimerResolution (
    UINT32                  *Resolution)
{
    ACPI_FUNCTION_TRACE (AcpiGetTimerResolution);


    if (!Resolution)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if ((AcpiGbl_FADT.Flags & ACPI_FADT_32BIT_TIMER) == 0)
    {
        *Resolution = 24;
    }
    else
    {
        *Resolution = 32;
    }

    return_ACPI_STATUS (AE_OK);
}

ACPI_EXPORT_SYMBOL (AcpiGetTimerResolution)


/******************************************************************************
 *
 * FUNCTION:    AcpiGetTimer
 *
 * PARAMETERS:  Ticks               - Where the timer value is returned
 *
 * RETURN:      Status and current timer value (ticks)
 *
 * DESCRIPTION: Obtains current value of ACPI PM Timer (in ticks).
 *
 *****************************************************************************/

ACPI_STATUS
AcpiGetTimer (
    UINT32                  *Ticks)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiGetTimer);


    if (!Ticks)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* ACPI 5.0A: PM Timer is optional */

    if (!AcpiGbl_FADT.XPmTimerBlock.Address)
    {
```

```
124         return_ACPI_STATUS (AE_SUPPORT);
125     }

127     Status = AcpiHwRead (Ticks, &AcpiGbl_FADT.XPmTimerBlock);

128     return_ACPI_STATUS (Status);
129 }

131 ACPI_EXPORT_SYMBOL (AcpiGetTimer)


134 /******************************************************************************
135  *
136  * FUNCTION:    AcpiGetTimerDuration
137  *
138  * PARAMETERS:  StartTicks          - Starting timestamp
139  *              EndTicks            - End timestamp
140  *              TimeElapsed         - Where the elapsed time is returned
141  *
142  * RETURN:      Status and TimeElapsed
143  *
144  * DESCRIPTION: Computes the time elapsed (in microseconds) between two
145  *              PM Timer time stamps, taking into account the possibility of
146  *              rollovers, the timer resolution, and timer frequency.
147  *
148  *              The PM Timer's clock ticks at roughly 3.6 times per
149  *              _microsecond_, and its clock continues through Cx state
150  *              transitions (unlike many CPU timestamp counters) -- making it
151  *              a versatile and accurate timer.
152  *
153  *              Note that this function accommodates only a single timer
154  *              rollover. Thus for 24-bit timers, this function should only
155  *              be used for calculating durations less than ~4.6 seconds
156  *              (~20 minutes for 32-bit timers) -- calculations below:
157  *
158  *              2**24 Ticks / 3,600,000 Ticks/Sec = 4.66 sec
159  *              2**32 Ticks / 3,600,000 Ticks/Sec = 1193 sec or 19.88 minutes
160  *
161  *****************************************************************************/

163 ACPI_STATUS
164 AcpiGetTimerDuration (
165     UINT32                  StartTicks,
166     UINT32                  EndTicks,
167     UINT32                  *TimeElapsed)
168 {
169     ACPI_STATUS             Status;
170     UINT32                  DeltaTicks;
171     UINT64                  Quotient;


174     ACPI_FUNCTION_TRACE (AcpiGetTimerDuration);


177     if (!TimeElapsed)
178     {
179         return_ACPI_STATUS (AE_BAD_PARAMETER);
180     }

182     /* ACPI 5.0A: PM Timer is optional */

184     if (!AcpiGbl_FADT.XPmTimerBlock.Address)
185     {
186         return_ACPI_STATUS (AE_SUPPORT);
187     }
```



```
189     /*
190      * Compute Tick Delta:
191      * Handle (max one) timer rollovers on 24-bit versus 32-bit timers.
192      */
193     if (StartTicks < EndTicks)
194     {
195         DeltaTicks = EndTicks - StartTicks;
196     }
197     else if (StartTicks > EndTicks)
198     {
199         if ((AcpiGbl_FADT.Flags & ACPI_FADT_32BIT_TIMER) == 0)
200         {
201             /* 24-bit Timer */

203             DeltaTicks = (((0x00FFFFFF - StartTicks) + EndTicks) & 0x00FFFFFF);
204         }
205         else
206         {
207             /* 32-bit Timer */

209             DeltaTicks = (0xFFFFFFFF - StartTicks) + EndTicks;
210         }
211     }
212     else /* StartTicks == EndTicks */
213     {
214         *TimeElapsed = 0;
215         return_ACPI_STATUS (AE_OK);
216     }

218     /*
219      * Compute Duration (Requires a 64-bit multiply and divide):
220      *
221      * TimeElapsed (microseconds) =
222      *  (DeltaTicks * ACPI_USEC_PER_SEC) / ACPI_PM_TIMER_FREQUENCY;
206      * TimeElapsed = (DeltaTicks * 1000000) / PM_TIMER_FREQUENCY;
223      */
224     Status = AcpiUtShortDivide (((UINT64) DeltaTicks) * ACPI_USEC_PER_SEC,
225                 ACPI_PM_TIMER_FREQUENCY, &Quotient, NULL);
208     Status = AcpiUtShortDivide (((UINT64) DeltaTicks) * 1000000,
209                 PM_TIMER_FREQUENCY, &Quotient, NULL);

227     *TimeElapsed = (UINT32) Quotient;
228     return_ACPI_STATUS (Status);
229 }

231 ACPI_EXPORT_SYMBOL (AcpiGetTimerDuration)

233 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
**********************************************************
   11591 Thu Dec 26 13:49:14 2013
new/usr/src/common/acpica/components/hardware/hwvalid.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: hwvalid - I/O request validation
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __HWVALID_C__

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwvalid")

/* Local prototypes */

static ACPI_STATUS
AcpiHwValidateIoRequest (
    ACPI_IO_ADDRESS         Address,
    UINT32                  BitWidth);
```



```
/*
 * Protected I/O ports. Some ports are always illegal, and some are
 * conditionally illegal. This table must remain ordered by port address.
 *
 * The table is used to implement the Microsoft port access rules that
 * first appeared in Windows XP. Some ports are always illegal, and some
 * ports are only illegal if the BIOS calls _OSI with a WinXP string or
 * later (meaning that the BIOS itelf is post-XP.)
 *
 * This provides ACPICA with the desired port protections and
 * Microsoft compatibility.
 *
 * Description of port entries:
 *  DMA:   DMA controller
 *  PIC0:  Programmable Interrupt Controller (8259A)
 *  PIT1:  System Timer 1
 *  PIT2:  System Timer 2 failsafe
 *  RTC:   Real-time clock
 *  CMOS:  Extended CMOS
 *  DMA1:  DMA 1 page registers
 *  DMA1L: DMA 1 Ch 0 low page
 *  DMA2:  DMA 2 page registers
 *  DMA2L: DMA 2 low page refresh
 *  ARBC:  Arbitration control
 *  SETUP: Reserved system board setup
 *  POS:   POS channel select
 *  PIC1:  Cascaded PIC
 *  IDMA:  ISA DMA
 *  ELCR:  PIC edge/level registers
 *  PCI:   PCI configuration space
 */
static const ACPI_PORT_INFO     AcpiProtectedPorts[] =
{
    {"DMA",     0x0000, 0x000F, ACPI_OSI_WIN_XP},
    {"PIC0",    0x0020, 0x0021, ACPI_ALWAYS_ILLEGAL},
    {"PIT1",    0x0040, 0x0043, ACPI_OSI_WIN_XP},
    {"PIT2",    0x0048, 0x004B, ACPI_OSI_WIN_XP},
    {"RTC",     0x0070, 0x0071, ACPI_OSI_WIN_XP},
    {"CMOS",    0x0074, 0x0076, ACPI_OSI_WIN_XP},
    {"DMA1",    0x0081, 0x0083, ACPI_OSI_WIN_XP},
    {"DMA1L",   0x0087, 0x0087, ACPI_OSI_WIN_XP},
    {"DMA2",    0x0089, 0x008B, ACPI_OSI_WIN_XP},
    {"DMA2L",   0x008F, 0x008F, ACPI_OSI_WIN_XP},
    {"ARBC",    0x0090, 0x0091, ACPI_OSI_WIN_XP},
    {"SETUP",   0x0093, 0x0094, ACPI_OSI_WIN_XP},
    {"POS",     0x0096, 0x0097, ACPI_OSI_WIN_XP},
    {"PIC1",    0x00A0, 0x00A1, ACPI_ALWAYS_ILLEGAL},
    {"IDMA",    0x00C0, 0x00DF, ACPI_OSI_WIN_XP},
    {"ELCR",    0x04D0, 0x04D1, ACPI_ALWAYS_ILLEGAL},
    {"PCI",     0x0CF8, 0x0CFF, ACPI_OSI_WIN_XP}
};

#define ACPI_PORT_INFO_ENTRIES  ACPI_ARRAY_LENGTH (AcpiProtectedPorts)


/******************************************************************************
 *
 * FUNCTION:    AcpiHwValidateIoRequest
 *
 * PARAMETERS:  Address             Address of I/O port/register
 *              BitWidth            Number of bits (8,16,32)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Validates an I/O request (address/length). Certain ports are
```

```
 125  *              always illegal and some ports are only illegal depending on
 126  *              the requests the BIOS AML code makes to the predefined
 127  *              _OSI method.
 128  *
 129  *****************************************************************************/

 131 static ACPI_STATUS
 132 AcpiHwValidateIoRequest (
 133     ACPI_IO_ADDRESS         Address,
 134     UINT32                  BitWidth)
 135 {
 136     UINT32                  i;
 137     UINT32                  ByteWidth;
 138     ACPI_IO_ADDRESS         LastAddress;
 139     const ACPI_PORT_INFO    *PortInfo;


 142     ACPI_FUNCTION_TRACE (HwValidateIoRequest);


 145     /* Supported widths are 8/16/32 */

 147     if ((BitWidth != 8) &&
 148         (BitWidth != 16) &&
 149         (BitWidth != 32))
 150     {
 151         ACPI_ERROR ((AE_INFO,
 152             "Bad BitWidth parameter: %8.8X", BitWidth));
 153         return (AE_BAD_PARAMETER);
 154     }

 156     PortInfo = AcpiProtectedPorts;
 157     ByteWidth = ACPI_DIV_8 (BitWidth);
 158     LastAddress = Address + ByteWidth - 1;

 160     ACPI_DEBUG_PRINT ((ACPI_DB_IO, "Address %p LastAddress %p Length %X",
 161         ACPI_CAST_PTR (void, Address), ACPI_CAST_PTR (void, LastAddress),
 162         ByteWidth));

 164     /* Maximum 16-bit address in I/O space */

 166     if (LastAddress > ACPI_UINT16_MAX)
 167     {
 168         ACPI_ERROR ((AE_INFO,
 169             "Illegal I/O port address/length above 64K: %p/0x%X",
 170             ACPI_CAST_PTR (void, Address), ByteWidth));
 171         return_ACPI_STATUS (AE_LIMIT);
 172     }

 174     /* Exit if requested address is not within the protected port table */

 176     if (Address > AcpiProtectedPorts[ACPI_PORT_INFO_ENTRIES - 1].End)
 177     {
 178         return_ACPI_STATUS (AE_OK);
 179     }

 181     /* Check request against the list of protected I/O ports */

 183     for (i = 0; i < ACPI_PORT_INFO_ENTRIES; i++, PortInfo++)
 184     {
 185         /*
 186          * Check if the requested address range will write to a reserved
 187          * port. Four cases to consider:
 188          *
 189          * 1) Address range is contained completely in the port address range
 190          * 2) Address range overlaps port range at the port range start
```



```
 191          * 3) Address range overlaps port range at the port range end
 192          * 4) Address range completely encompasses the port range
 193          */
 194         if ((Address <= PortInfo->End) && (LastAddress >= PortInfo->Start))
 195         {
 196             /* Port illegality may depend on the _OSI calls made by the BIOS */

 198             if (AcpiGbl_OsiData >= PortInfo->OsiDependency)
 199             {
 200                 ACPI_DEBUG_PRINT ((ACPI_DB_IO,
 201                     "Denied AML access to port 0x%p/%X (%s 0x%.4X-0x%.4X)",
 202                     ACPI_CAST_PTR (void, Address), ByteWidth, PortInfo->Name,
 203                     PortInfo->Start, PortInfo->End));

 205                 return_ACPI_STATUS (AE_AML_ILLEGAL_ADDRESS);
 206             }
 207         }

 209         /* Finished if address range ends before the end of this port */

 211         if (LastAddress <= PortInfo->End)
 212         {
 213             break;
 214         }
 215     }

 217     return_ACPI_STATUS (AE_OK);
 218 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   19617 Thu Dec 26 13:49:15 2013
new/usr/src/common/acpica/components/hardware/hwxface.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: hwxface - Public ACPICA hardware interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwxface")


/******************************************************************************
 *
 * FUNCTION:    AcpiReset
 *
```



```
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Set reset register in memory or IO space. Note: Does not
 *              support reset register in PCI config space, this must be
 *              handled separately.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiReset (
    void)
{
    ACPI_GENERIC_ADDRESS    *ResetReg;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiReset);


    ResetReg = &AcpiGbl_FADT.ResetRegister;

    /* Check if the reset register is supported */

    if (!(AcpiGbl_FADT.Flags & ACPI_FADT_RESET_REGISTER) ||
        !ResetReg->Address)
    {
        return_ACPI_STATUS (AE_NOT_EXIST);
    }

    if (ResetReg->SpaceId == ACPI_ADR_SPACE_SYSTEM_IO)
    {
        /*
         * For I/O space, write directly to the OSL. This bypasses the port
         * validation mechanism, which may block a valid write to the reset
         * register.
         *
         * NOTE:
         * The ACPI spec requires the reset register width to be 8, so we
         * hardcode it here and ignore the FADT value. This maintains
         * compatibility with other ACPI implementations that have allowed
         * BIOS code with bad register width values to go unnoticed.
         */
        Status = AcpiOsWritePort ((ACPI_IO_ADDRESS) ResetReg->Address,
            AcpiGbl_FADT.ResetValue, ACPI_RESET_REGISTER_WIDTH);
                AcpiGbl_FADT.ResetValue, ResetReg->BitWidth);
    }
    else
    {
        /* Write the reset value to the reset register */

        Status = AcpiHwWrite (AcpiGbl_FADT.ResetValue, ResetReg);
    }

    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiReset)


/******************************************************************************
 *
 * FUNCTION:    AcpiRead
 *
 * PARAMETERS:  Value               - Where the value is returned
```

```
 *              Reg                 - GAS register structure
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Read from either memory or IO space.
 *
 * LIMITATIONS: <These limitations also apply to AcpiWrite>
 *      BitWidth must be exactly 8, 16, 32, or 64.
 *      SpaceID must be SystemMemory or SystemIO.
 *      BitOffset and AccessWidth are currently ignored, as there has
 *          not been a need to implement these.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiRead (
    UINT64                  *ReturnValue,
    ACPI_GENERIC_ADDRESS    *Reg)
{
    UINT32                  ValueLo;
    UINT32                  ValueHi;
    UINT32                  Value;
    UINT32                  Width;
    UINT64                  Address;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_NAME (AcpiRead);


    if (!ReturnValue)
    {
        return (AE_BAD_PARAMETER);
    }

    /* Validate contents of the GAS register. Allow 64-bit transfers */

    Status = AcpiHwValidateRegister (Reg, 64, &Address);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    Width = Reg->BitWidth;
    if (Width == 64)
    {
        Width = 32; /* Break into two 32-bit transfers */
    }

    /* Initialize entire 64-bit return value to zero */

    *ReturnValue = 0;
    Value = 0;

    /*
     * Two address spaces supported: Memory or I/O. PCI_Config is
     * Two address spaces supported: Memory or IO. PCI_Config is
     * not supported here because the GAS structure is insufficient
     */
    if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
    {
        Status = AcpiOsReadMemory ((ACPI_PHYSICAL_ADDRESS)
                    Address, ReturnValue, Reg->BitWidth);
                    Address, &Value, Width);
        if (ACPI_FAILURE (Status))
        {
```



```
            return (Status);
        }
    }
    else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
        *ReturnValue = Value;

        if (Reg->BitWidth == 64)
    {
        ValueLo = 0;
        ValueHi = 0;
            /* Read the top 32 bits */

        Width = Reg->BitWidth;
        if (Width == 64)
            Status = AcpiOsReadMemory ((ACPI_PHYSICAL_ADDRESS)
                        (Address + 4), &Value, 32);
            if (ACPI_FAILURE (Status))
        {
            Width = 32; /* Break into two 32-bit transfers */
                return (Status);
        }

            *ReturnValue |= ((UINT64) Value << 32);
        }
    }
    else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
    {
        Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
                    Address, &ValueLo, Width);
                    Address, &Value, Width);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }
        *ReturnValue = Value;

        if (Reg->BitWidth == 64)
        {
            /* Read the top 32 bits */

            Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
                        (Address + 4), &ValueHi, 32);
                        (Address + 4), &Value, 32);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
            *ReturnValue |= ((UINT64) Value << 32);
        }

        /* Set the return value only if status is AE_OK */

        *ReturnValue = (ValueLo | ((UINT64) ValueHi << 32));
    }

    ACPI_DEBUG_PRINT ((ACPI_DB_IO,
        "Read:  %8.8X%8.8X width %2d from %8.8X%8.8X (%s)\n",
        ACPI_FORMAT_UINT64 (*ReturnValue), Reg->BitWidth,
        ACPI_FORMAT_UINT64 (Address),
        AcpiUtGetRegionName (Reg->SpaceId)));

    return (AE_OK);
    return (Status);
}

ACPI_EXPORT_SYMBOL (AcpiRead)
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiWrite
 *
 * PARAMETERS:  Value               - Value to be written
 *              Reg                 - GAS register structure
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Write to either memory or IO space.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiWrite (
    UINT64                  Value,
    ACPI_GENERIC_ADDRESS    *Reg)
{
    UINT32                  Width;
    UINT64                  Address;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_NAME (AcpiWrite);


    /* Validate contents of the GAS register. Allow 64-bit transfers */

    Status = AcpiHwValidateRegister (Reg, 64, &Address);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    Width = Reg->BitWidth;
    if (Width == 64)
    {
        Width = 32; /* Break into two 32-bit transfers */
    }

    /*
     * Two address spaces supported: Memory or IO. PCI_Config is
     * not supported here because the GAS structure is insufficient
     */
    if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
    {
        Status = AcpiOsWriteMemory ((ACPI_PHYSICAL_ADDRESS)
                    Address, Value, Reg->BitWidth);
                    Address, ACPI_LODWORD (Value), Width);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }
    }
    else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */


        if (Reg->BitWidth == 64)
    {
        Width = Reg->BitWidth;
        if (Width == 64)
            Status = AcpiOsWriteMemory ((ACPI_PHYSICAL_ADDRESS)
                        (Address + 4), ACPI_HIDWORD (Value), 32);
            if (ACPI_FAILURE (Status))
        {
```



```
            Width = 32; /* Break into two 32-bit transfers */
                return (Status);
        }

        }
    }
    else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
    {
        Status = AcpiHwWritePort ((ACPI_IO_ADDRESS)
                    Address, ACPI_LODWORD (Value), Width);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        if (Reg->BitWidth == 64)
        {
            Status = AcpiHwWritePort ((ACPI_IO_ADDRESS)
                        (Address + 4), ACPI_HIDWORD (Value), 32);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
        }
    }

    ACPI_DEBUG_PRINT ((ACPI_DB_IO,
        "Wrote: %8.8X%8.8X width %2d   to %8.8X%8.8X (%s)\n",
        ACPI_FORMAT_UINT64 (Value), Reg->BitWidth,
        ACPI_FORMAT_UINT64 (Address),
        AcpiUtGetRegionName (Reg->SpaceId)));

    return (Status);
}

ACPI_EXPORT_SYMBOL (AcpiWrite)


#if (!ACPI_REDUCED_HARDWARE)
/*******************************************************************************
 *
 * FUNCTION:    AcpiReadBitRegister
 *
 * PARAMETERS:  RegisterId      - ID of ACPI Bit Register to access
 *              ReturnValue     - Value that was read from the register,
 *                                normalized to bit position zero.
 *
 * RETURN:      Status and the value read from the specified Register. Value
 *              returned is normalized to bit0 (is shifted all the way right)
 *
 * DESCRIPTION: ACPI BitRegister read function. Does not acquire the HW lock.
 *
 * SUPPORTS:    Bit fields in PM1 Status, PM1 Enable, PM1 Control, and
 *              PM2 Control.
 *
 * Note: The hardware lock is not required when reading the ACPI bit registers
 *       since almost all of them are single bit and it does not matter that
 *       the parent hardware register can be split across two physical
 *       registers. The only multi-bit field is SLP_TYP in the PM1 control
 *       register, but this field does not cross an 8-bit boundary (nor does
 *       it make much sense to actually read this field.)
 *
 ******************************************************************************/

ACPI_STATUS
AcpiReadBitRegister (
```

```
338     UINT32                  RegisterId,
339     UINT32                  *ReturnValue)
340 {
341     ACPI_BIT_REGISTER_INFO  *BitRegInfo;
342     UINT32                  RegisterValue;
343     UINT32                  Value;
344     ACPI_STATUS             Status;


347     ACPI_FUNCTION_TRACE_U32 (AcpiReadBitRegister, RegisterId);


350     /* Get the info structure corresponding to the requested ACPI Register */

352     BitRegInfo = AcpiHwGetBitRegisterInfo (RegisterId);
353     if (!BitRegInfo)
354     {
355         return_ACPI_STATUS (AE_BAD_PARAMETER);
356     }

358     /* Read the entire parent register */

360     Status = AcpiHwRegisterRead (BitRegInfo->ParentRegister,
361                 &RegisterValue);
362     if (ACPI_FAILURE (Status))
363     {
364         return_ACPI_STATUS (Status);
365     }

367     /* Normalize the value that was read, mask off other bits */

369     Value = ((RegisterValue & BitRegInfo->AccessBitMask)
370                 >> BitRegInfo->BitPosition);

372     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
373         "BitReg %X, ParentReg %X, Actual %8.8X, ReturnValue %8.8X\n",
374         RegisterId, BitRegInfo->ParentRegister, RegisterValue, Value));

376     *ReturnValue = Value;
377     return_ACPI_STATUS (AE_OK);
378 }

380 ACPI_EXPORT_SYMBOL (AcpiReadBitRegister)


383 /******************************************************************************
384  *
385  * FUNCTION:    AcpiWriteBitRegister
386  *
387  * PARAMETERS:  RegisterId      - ID of ACPI Bit Register to access
388  *              Value           - Value to write to the register, in bit
389  *                                position zero. The bit is automatically
390  *                                position zero. The bit is automaticallly
391  *                                shifted to the correct position.
392  *
393  * RETURN:      Status
394  *
395  * DESCRIPTION: ACPI Bit Register write function. Acquires the hardware lock
396  *              since most operations require a read/modify/write sequence.
397  *
398  * SUPPORTS:    Bit fields in PM1 Status, PM1 Enable, PM1 Control, and
399  *              PM2 Control.
400  *
401  * Note that at this level, the fact that there may be actually two
402  * hardware registers (A and B - and B may not exist) is abstracted.
```



```
403  ******************************************************************************/

405 ACPI_STATUS
406 AcpiWriteBitRegister (
407     UINT32                  RegisterId,
408     UINT32                  Value)
409 {
410     ACPI_BIT_REGISTER_INFO  *BitRegInfo;
411     ACPI_CPU_FLAGS          LockFlags;
412     UINT32                  RegisterValue;
413     ACPI_STATUS             Status = AE_OK;


416     ACPI_FUNCTION_TRACE_U32 (AcpiWriteBitRegister, RegisterId);


419     /* Get the info structure corresponding to the requested ACPI Register */

421     BitRegInfo = AcpiHwGetBitRegisterInfo (RegisterId);
422     if (!BitRegInfo)
423     {
424         return_ACPI_STATUS (AE_BAD_PARAMETER);
425     }

427     LockFlags = AcpiOsAcquireLock (AcpiGbl_HardwareLock);

429     /*
430      * At this point, we know that the parent register is one of the
431      * following: PM1 Status, PM1 Enable, PM1 Control, or PM2 Control
432      */
433     if (BitRegInfo->ParentRegister != ACPI_REGISTER_PM1_STATUS)
434     {
435         /*
436          * 1) Case for PM1 Enable, PM1 Control, and PM2 Control
437          *
438          * Perform a register read to preserve the bits that we are not
439          * interested in
440          */
441         Status = AcpiHwRegisterRead (BitRegInfo->ParentRegister,
442                     &RegisterValue);
443         if (ACPI_FAILURE (Status))
444         {
445             goto UnlockAndExit;
446         }

448         /*
449          * Insert the input bit into the value that was just read
450          * and write the register
451          */
452         ACPI_REGISTER_INSERT_VALUE (RegisterValue, BitRegInfo->BitPosition,
453             BitRegInfo->AccessBitMask, Value);

455         Status = AcpiHwRegisterWrite (BitRegInfo->ParentRegister,
456                     RegisterValue);
457     }
458     else
459     {
460         /*
461          * 2) Case for PM1 Status
462          *
463          * The Status register is different from the rest. Clear an event
464          * by writing 1, writing 0 has no effect. So, the only relevant
465          * information is the single bit we're interested in, all others
466          * should be written as 0 so they will be left unchanged.
467          */
468         RegisterValue = ACPI_REGISTER_PREPARE_BITS (Value,
```

```
469             BitRegInfo->BitPosition, BitRegInfo->AccessBitMask);

471         /* No need to write the register if value is all zeros */

473         if (RegisterValue)
474         {
475             Status = AcpiHwRegisterWrite (ACPI_REGISTER_PM1_STATUS,
476                         RegisterValue);
477         }
478     }

480     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
481         "BitReg %X, ParentReg %X, Value %8.8X, Actual %8.8X\n",
482         RegisterId, BitRegInfo->ParentRegister, Value, RegisterValue));


485 UnlockAndExit:

487     AcpiOsReleaseLock (AcpiGbl_HardwareLock, LockFlags);
488     return_ACPI_STATUS (Status);
489 }

491 ACPI_EXPORT_SYMBOL (AcpiWriteBitRegister)

493 #endif /* !ACPI_REDUCED_HARDWARE */


496 /*******************************************************************************
497  *
498  * FUNCTION:    AcpiGetSleepTypeData
499  *
500  * PARAMETERS:  SleepState          - Numeric sleep state
501  *              *SleepTypeA         - Where SLP_TYPa is returned
502  *              *SleepTypeB         - Where SLP_TYPb is returned
503  *
504  * RETURN:      Status
517  * RETURN:      Status - ACPI status
505  *
506  * DESCRIPTION: Obtain the SLP_TYPa and SLP_TYPb values for the requested
507  *              sleep state via the appropriate \_Sx object.
519  * DESCRIPTION: Obtain the SLP_TYPa and SLP_TYPb values for the requested sleep
520  *              state.
508  *
509  *  The sleep state package returned from the corresponding \_Sx_ object
510  *  must contain at least one integer.
511  *
512  *  March 2005:
513  *  Added support for a package that contains two integers. This
514  *  goes against the ACPI specification which defines this object as a
515  *  package with one encoded DWORD integer. However, existing practice
516  *  by many BIOS vendors is to return a package with 2 or more integer
517  *  elements, at least one per sleep type (A/B).
518  *
519  *  January 2013:
520  *  Therefore, we must be prepared to accept a package with either a
521  *  single integer or multiple integers.
522  *
523  *  The single integer DWORD format is as follows:
524  *      BYTE 0 - Value for the PM1A SLP_TYP register
525  *      BYTE 1 - Value for the PM1B SLP_TYP register
526  *      BYTE 2-3 - Reserved
527  *
528  *  The dual integer format is as follows:
529  *      Integer 0 - Value for the PM1A SLP_TYP register
530  *      Integer 1 - Value for the PM1A SLP_TYP register
531  *
```



```
532  ******************************************************************************/

534 ACPI_STATUS
535 AcpiGetSleepTypeData (
536     UINT8                   SleepState,
537     UINT8                   *SleepTypeA,
538     UINT8                   *SleepTypeB)
539 {
540     ACPI_STATUS             Status;
530     ACPI_STATUS             Status = AE_OK;
541     ACPI_EVALUATE_INFO      *Info;
542     ACPI_OPERAND_OBJECT     **Elements;


545     ACPI_FUNCTION_TRACE (AcpiGetSleepTypeData);


548     /* Validate parameters */

550     if ((SleepState > ACPI_S_STATES_MAX) ||
551         !SleepTypeA || !SleepTypeB)
540         !SleepTypeA ||
541         !SleepTypeB)
552     {
553         return_ACPI_STATUS (AE_BAD_PARAMETER);
554     }

556     /* Allocate the evaluation information block */

558     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
559     if (!Info)
560     {
561         return_ACPI_STATUS (AE_NO_MEMORY);
562     }

564     /*
565      * Evaluate the \_Sx namespace object containing the register values
566      * for this state
567      */
568     Info->RelativePathname = ACPI_CAST_PTR (
569         char, AcpiGbl_SleepStateNames[SleepState]);
554     Info->Pathname = ACPI_CAST_PTR (char, AcpiGbl_SleepStateNames[SleepState]);

556     /* Evaluate the namespace object containing the values for this state */

570     Status = AcpiNsEvaluate (Info);
571     if (ACPI_FAILURE (Status))
572     {
561         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
562             "%s while evaluating SleepState [%s]\n",
563             AcpiFormatException (Status), Info->Pathname));

573         goto Cleanup;
574     }

576     /* Must have a return object */

578     if (!Info->ReturnObject)
579     {
580         ACPI_ERROR ((AE_INFO, "No Sleep State object returned from [%s]",
581             Info->RelativePathname));
582         Status = AE_AML_NO_RETURN_VALUE;
583         goto Cleanup;
573             Info->Pathname));
574         Status = AE_NOT_EXIST;
584     }
```

```
586     /* Return object must be of type Package */
577     /* It must be of type Package */

588     if (Info->ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
579     else if (Info->ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
589     {
590         ACPI_ERROR ((AE_INFO, "Sleep State return object is not a Package"));
591         Status = AE_AML_OPERAND_TYPE;
592         goto Cleanup1;
593     }

595     /*
596      * Any warnings about the package length or the object types have
597      * already been issued by the predefined name module -- there is no
598      * need to repeat them here.
586      * The package must have at least two elements. NOTE (March 2005): This
587      * goes against the current ACPI spec which defines this object as a
588      * package with one encoded DWORD element. However, existing practice
589      * by BIOS vendors seems to be to have 2 or more elements, at least
590      * one per sleep type (A/B).
599      */
600     Elements = Info->ReturnObject->Package.Elements;
601     switch (Info->ReturnObject->Package.Count)
592     else if (Info->ReturnObject->Package.Count < 2)
602     {
603     case 0:

605         Status = AE_AML_PACKAGE_LIMIT;
606         break;

608     case 1:

610         if (Elements[0]->Common.Type != ACPI_TYPE_INTEGER)
611         {
612             Status = AE_AML_OPERAND_TYPE;
613             break;
594         ACPI_ERROR ((AE_INFO,
595             "Sleep State return package does not have at least two elements"));
596         Status = AE_AML_NO_OPERAND;
614         }

616         /* A valid _Sx_ package with one integer */
599     /* The first two elements must both be of type Integer */

618         *SleepTypeA = (UINT8) Elements[0]->Integer.Value;
619         *SleepTypeB = (UINT8) (Elements[0]->Integer.Value >> 8);
620         break;

622     case 2:
623     default:

625         if ((Elements[0]->Common.Type != ACPI_TYPE_INTEGER) ||
626             (Elements[1]->Common.Type != ACPI_TYPE_INTEGER))
601     else if (((Info->ReturnObject->Package.Elements[0])->Common.Type
602                 != ACPI_TYPE_INTEGER) ||
603              ((Info->ReturnObject->Package.Elements[1])->Common.Type
604                 != ACPI_TYPE_INTEGER))
627         {
606         ACPI_ERROR ((AE_INFO,
607             "Sleep State return package elements are not both Integers "
608             "(%s, %s)",
609             AcpiUtGetObjectTypeName (Info->ReturnObject->Package.Elements[0]),
610             AcpiUtGetObjectTypeName (Info->ReturnObject->Package.Elements[1])));
628             Status = AE_AML_OPERAND_TYPE;
629             break;
```



```
630         }
613     else
614     {
615         /* Valid _Sx_ package size, type, and value */

632         /* A valid _Sx_ package with two integers */

634         *SleepTypeA = (UINT8) Elements[0]->Integer.Value;
635         *SleepTypeB = (UINT8) Elements[1]->Integer.Value;
636         break;
617         *SleepTypeA = (UINT8)
618             (Info->ReturnObject->Package.Elements[0])->Integer.Value;
619         *SleepTypeB = (UINT8)
620             (Info->ReturnObject->Package.Elements[1])->Integer.Value;
637     }

639 Cleanup1:
640     AcpiUtRemoveReference (Info->ReturnObject);

642 Cleanup:
643     if (ACPI_FAILURE (Status))
644     {
645         ACPI_EXCEPTION ((AE_INFO, Status,
646             "While evaluating Sleep State [%s]", Info->RelativePathname));
626             "While evaluating SleepState [%s], bad Sleep object %p type %s",
627             Info->Pathname, Info->ReturnObject,
628             AcpiUtGetObjectTypeName (Info->ReturnObject)));
647     }

631     AcpiUtRemoveReference (Info->ReturnObject);

633 Cleanup:
649     ACPI_FREE (Info);
650     return_ACPI_STATUS (Status);
651 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   13990 Thu Dec 26 13:49:15 2013
new/usr/src/common/acpica/components/hardware/hwxfsleep.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Name: hwxfsleep.c - ACPI Hardware Sleep/Wake External Interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_HARDWARE
        ACPI_MODULE_NAME    ("hwxfsleep")

/* Local prototypes */

static ACPI_STATUS
AcpiHwSleepDispatch (
    UINT8                   SleepState,
    UINT32                  FunctionId);

/*
 * Dispatch table used to efficiently branch to the various sleep
```



```
 * functions.
 */
#define ACPI_SLEEP_FUNCTION_ID          0
#define ACPI_WAKE_PREP_FUNCTION_ID      1
#define ACPI_WAKE_FUNCTION_ID           2

/* Legacy functions are optional, based upon ACPI_REDUCED_HARDWARE */

static ACPI_SLEEP_FUNCTIONS         AcpiSleepDispatch[] =
{
    {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacySleep),    AcpiHwExtendedSleep},
    {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacyWakePrep), AcpiHwExtendedWakePrep},
    {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacyWake),     AcpiHwExtendedWake}
};


/*
 * These functions are removed for the ACPI_REDUCED_HARDWARE case:
 *      AcpiSetFirmwareWakingVector
 *      AcpiSetFirmwareWakingVector64
 *      AcpiEnterSleepStateS4bios
 */

#if (!ACPI_REDUCED_HARDWARE)
/*******************************************************************************
 *
 * FUNCTION:    AcpiSetFirmwareWakingVector
 *
 * PARAMETERS:  PhysicalAddress     - 32-bit physical address of ACPI real mode
 *                                    entry point.
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Sets the 32-bit FirmwareWakingVector field of the FACS
 *
 ******************************************************************************/

ACPI_STATUS
AcpiSetFirmwareWakingVector (
    UINT32                  PhysicalAddress)
{
    ACPI_FUNCTION_TRACE (AcpiSetFirmwareWakingVector);


    /*
     * According to the ACPI specification 2.0c and later, the 64-bit
     * waking vector should be cleared and the 32-bit waking vector should
     * be used, unless we want the wake-up code to be called by the BIOS in
     * Protected Mode. Some systems (for example HP dv5-1004nr) are known
     * to fail to resume if the 64-bit vector is used.
     */

    /* Set the 32-bit vector */

    AcpiGbl_FACS->FirmwareWakingVector = PhysicalAddress;

    /* Clear the 64-bit vector if it exists */

    if ((AcpiGbl_FACS->Length > 32) && (AcpiGbl_FACS->Version >= 1))
    {
        AcpiGbl_FACS->XFirmwareWakingVector = 0;
    }

    return_ACPI_STATUS (AE_OK);
}
```

```
127 ACPI_EXPORT_SYMBOL (AcpiSetFirmwareWakingVector)


130 #if ACPI_MACHINE_WIDTH == 64
131 /*******************************************************************************
132  *
133  * FUNCTION:    AcpiSetFirmwareWakingVector64
134  *
135  * PARAMETERS:  PhysicalAddress     - 64-bit physical address of ACPI protected
136  *                                    mode entry point.
137  *
138  * RETURN:      Status
139  *
140  * DESCRIPTION: Sets the 64-bit X_FirmwareWakingVector field of the FACS, if
141  *              it exists in the table. This function is intended for use with
142  *              64-bit host operating systems.
143  *
144  ******************************************************************************/

146 ACPI_STATUS
147 AcpiSetFirmwareWakingVector64 (
148     UINT64                  PhysicalAddress)
149 {
150     ACPI_FUNCTION_TRACE (AcpiSetFirmwareWakingVector64);


153     /* Determine if the 64-bit vector actually exists */

155     if ((AcpiGbl_FACS->Length <= 32) || (AcpiGbl_FACS->Version < 1))
156     {
157         return_ACPI_STATUS (AE_NOT_EXIST);
158     }

160     /* Clear 32-bit vector, set the 64-bit X_ vector */

162     AcpiGbl_FACS->FirmwareWakingVector = 0;
163     AcpiGbl_FACS->XFirmwareWakingVector = PhysicalAddress;
164     return_ACPI_STATUS (AE_OK);
165 }

167 ACPI_EXPORT_SYMBOL (AcpiSetFirmwareWakingVector64)
168 #endif


171 /*******************************************************************************
172  *
173  * FUNCTION:    AcpiEnterSleepStateS4bios
174  *
175  * PARAMETERS:  None
176  *
177  * RETURN:      Status
178  *
179  * DESCRIPTION: Perform a S4 bios request.
180  *              THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
181  *
182  ******************************************************************************/

184 ACPI_STATUS
185 AcpiEnterSleepStateS4bios (
186     void)
187 {
188     UINT32                  InValue;
189     ACPI_STATUS             Status;


192     ACPI_FUNCTION_TRACE (AcpiEnterSleepStateS4bios);
```



```


195     /* Clear the wake status bit (PM1) */

197     Status = AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, ACPI_CLEAR_STATUS);
198     if (ACPI_FAILURE (Status))
199     {
200         return_ACPI_STATUS (Status);
201     }

203     Status = AcpiHwClearAcpiStatus ();
204     if (ACPI_FAILURE (Status))
205     {
206         return_ACPI_STATUS (Status);
207     }

209     /*
210      * 1) Disable/Clear all GPEs
211      * 2) Enable all wakeup GPEs
212      */
213     Status = AcpiHwDisableAllGpes ();
214     if (ACPI_FAILURE (Status))
215     {
216         return_ACPI_STATUS (Status);
217     }
218     AcpiGbl_SystemAwakeAndRunning = FALSE;

220     Status = AcpiHwEnableAllWakeupGpes ();
221     if (ACPI_FAILURE (Status))
222     {
223         return_ACPI_STATUS (Status);
224     }

226     ACPI_FLUSH_CPU_CACHE ();

228     Status = AcpiHwWritePort (AcpiGbl_FADT.SmiCommand,
229                 (UINT32) AcpiGbl_FADT.S4BiosRequest, 8);

231     do {
232         AcpiOsStall (ACPI_USEC_PER_MSEC);
233         Status = AcpiReadBitRegister (ACPI_BITREG_WAKE_STATUS, &InValue);
234         if (ACPI_FAILURE (Status))
235         {
236             return_ACPI_STATUS (Status);
237         }
238     } while (!InValue);

240     return_ACPI_STATUS (AE_OK);
241 }

243 ACPI_EXPORT_SYMBOL (AcpiEnterSleepStateS4bios)

245 #endif /* !ACPI_REDUCED_HARDWARE */


248 /*******************************************************************************
249  *
250  * FUNCTION:    AcpiHwSleepDispatch
251  *
252  * PARAMETERS:  SleepState          - Which sleep state to enter/exit
253  *              FunctionId          - Sleep, WakePrep, or Wake
254  *
255  * RETURN:      Status from the invoked sleep handling function.
256  *
257  * DESCRIPTION: Dispatch a sleep/wake request to the appropriate handling
258  *              function.
```

```
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiHwSleepDispatch (
    UINT8                   SleepState,
    UINT32                  FunctionId)
{
    ACPI_STATUS             Status;
    ACPI_SLEEP_FUNCTIONS    *SleepFunctions = &AcpiSleepDispatch[FunctionId];


#if (!ACPI_REDUCED_HARDWARE)
    /*
     * If the Hardware Reduced flag is set (from the FADT), we must
     * use the extended sleep registers (FADT). Note: As per the ACPI
     * specification, these extended registers are to be used for HW-reduced
     * platforms only. They are not general-purpose replacements for the
     * legacy PM register sleep support.
     */
    if (AcpiGbl_ReducedHardware)
    {
        Status = SleepFunctions->ExtendedFunction (SleepState);
    }
    else
    {
        /* Legacy sleep */

        Status = SleepFunctions->LegacyFunction (SleepState);
    }

    return (Status);

#else
    /*
     * For the case where reduced-hardware-only code is being generated,
     * we know that only the extended sleep registers are available
     */
    Status = SleepFunctions->ExtendedFunction (SleepState);
    return (Status);

#endif /* !ACPI_REDUCED_HARDWARE */
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiEnterSleepStatePrep
 *
 * PARAMETERS:  SleepState          - Which sleep state to enter
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Prepare to enter a system sleep state.
 *              This function must execute with interrupts enabled.
 *              We break sleeping into 2 stages so that OSPM can handle
 *              various OS-specific tasks between the two steps.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEnterSleepStatePrep (
    UINT8                   SleepState)
{
    ACPI_STATUS             Status;
    ACPI_OBJECT_LIST        ArgList;
```



```
    ACPI_OBJECT             Arg;
    UINT32                  SstValue;


    ACPI_FUNCTION_TRACE (AcpiEnterSleepStatePrep);


    Status = AcpiGetSleepTypeData (SleepState,
                    &AcpiGbl_SleepTypeA, &AcpiGbl_SleepTypeB);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Execute the _PTS method (Prepare To Sleep) */

    ArgList.Count = 1;
    ArgList.Pointer = &Arg;
    Arg.Type = ACPI_TYPE_INTEGER;
    Arg.Integer.Value = SleepState;

    Status = AcpiEvaluateObject (NULL, METHOD_PATHNAME__PTS, &ArgList, NULL);
    if (ACPI_FAILURE (Status) && Status != AE_NOT_FOUND)
    {
        return_ACPI_STATUS (Status);
    }

    /* Setup the argument to the _SST method (System STatus) */

    switch (SleepState)
    {
    case ACPI_STATE_S0:

        SstValue = ACPI_SST_WORKING;
        break;

    case ACPI_STATE_S1:
    case ACPI_STATE_S2:
    case ACPI_STATE_S3:

        SstValue = ACPI_SST_SLEEPING;
        break;

    case ACPI_STATE_S4:

        SstValue = ACPI_SST_SLEEP_CONTEXT;
        break;

    default:

        SstValue = ACPI_SST_INDICATOR_OFF; /* Default is off */
        break;
    }

    /*
     * Set the system indicators to show the desired sleep state.
     * _SST is an optional method (return no error if not found)
     */
    AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, SstValue);
    return_ACPI_STATUS (AE_OK);
}

ACPI_EXPORT_SYMBOL (AcpiEnterSleepStatePrep)


/*******************************************************************************
```

```
 *
 * FUNCTION:    AcpiEnterSleepState
 *
 * PARAMETERS:  SleepState          - Which sleep state to enter
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Enter a system sleep state
 *              THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEnterSleepState (
    UINT8                   SleepState)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiEnterSleepState);


    if ((AcpiGbl_SleepTypeA > ACPI_SLEEP_TYPE_MAX) ||
        (AcpiGbl_SleepTypeB > ACPI_SLEEP_TYPE_MAX))
    {
        ACPI_ERROR ((AE_INFO, "Sleep values out of range: A=0x%X B=0x%X",
            AcpiGbl_SleepTypeA, AcpiGbl_SleepTypeB));
        return_ACPI_STATUS (AE_AML_OPERAND_VALUE);
    }

    Status = AcpiHwSleepDispatch (SleepState, ACPI_SLEEP_FUNCTION_ID);
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiEnterSleepState)


/******************************************************************************
 *
 * FUNCTION:    AcpiLeaveSleepStatePrep
 *
 * PARAMETERS:  SleepState          - Which sleep state we are exiting
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform the first state of OS-independent ACPI cleanup after a
 *              sleep. Called with interrupts DISABLED.
 *              We break wake/resume into 2 stages so that OSPM can handle
 *              various OS-specific tasks between the two steps.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiLeaveSleepStatePrep (
    UINT8                   SleepState)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiLeaveSleepStatePrep);


    Status = AcpiHwSleepDispatch (SleepState, ACPI_WAKE_PREP_FUNCTION_ID);
    return_ACPI_STATUS (Status);
}
```



```
ACPI_EXPORT_SYMBOL (AcpiLeaveSleepStatePrep)


/******************************************************************************
 *
 * FUNCTION:    AcpiLeaveSleepState
 *
 * PARAMETERS:  SleepState          - Which sleep state we are exiting
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform OS-independent ACPI cleanup after a sleep
 *              Called with interrupts ENABLED.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiLeaveSleepState (
    UINT8                   SleepState)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiLeaveSleepState);


    Status = AcpiHwSleepDispatch (SleepState, ACPI_WAKE_FUNCTION_ID);
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL (AcpiLeaveSleepState)
```

****
22575 Thu Dec 26 13:49:15 2013
new/usr/src/common/acpica/components/namespace/nsaccess.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: nsaccess - Top-level functions for accessing ACPI namespace
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __NSACCESS_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "amlcode.h"
  49 #include "acnamesp.h"
  50 #include "acdispat.h"


  53 #define _COMPONENT          ACPI_NAMESPACE
  54         ACPI_MODULE_NAME    ("nsaccess")


  57 /*******************************************************************************
  58  *
  59  * FUNCTION:    AcpiNsRootInitialize
```



```
  60  *
  61  * PARAMETERS:  None
  62  *
  63  * RETURN:      Status
  64  *
  65  * DESCRIPTION: Allocate and initialize the default root named objects
  66  *
  67  * MUTEX:       Locks namespace for entire execution
  68  *
  69  ******************************************************************************/

  71 ACPI_STATUS
  72 AcpiNsRootInitialize (
  73     void)
  74 {
  75     ACPI_STATUS                 Status;
  76     const ACPI_PREDEFINED_NAMES *InitVal = NULL;
  77     ACPI_NAMESPACE_NODE         *NewNode;
  78     ACPI_OPERAND_OBJECT         *ObjDesc;
  79     ACPI_STRING                 Val = NULL;


  82     ACPI_FUNCTION_TRACE (NsRootInitialize);


  85     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
  86     if (ACPI_FAILURE (Status))
  87     {
  88         return_ACPI_STATUS (Status);
  89     }

  91     /*
  92      * The global root ptr is initially NULL, so a non-NULL value indicates
  93      * that AcpiNsRootInitialize() has already been called; just return.
  94      */
  95     if (AcpiGbl_RootNode)
  96     {
  97         Status = AE_OK;
  98         goto UnlockAndExit;
  99     }

 101     /*
 102      * Tell the rest of the subsystem that the root is initialized
 103      * (This is OK because the namespace is locked)
 104      */
 105     AcpiGbl_RootNode = &AcpiGbl_RootNodeStruct;

 107     /* Enter the pre-defined names in the name table */

 109     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
 110         "Entering predefined entries into namespace\n"));

 112     for (InitVal = AcpiGbl_PreDefinedNames; InitVal->Name; InitVal++)
 113     {
 114         /* _OSI is optional for now, will be permanent later */

 116         if (!ACPI_STRCMP (InitVal->Name, "_OSI") && !AcpiGbl_CreateOsiMethod)
 117         {
 118             continue;
 119         }

 121         Status = AcpiNsLookup (NULL, InitVal->Name, InitVal->Type,
 122                         ACPI_IMODE_LOAD_PASS2, ACPI_NS_NO_UPSEARCH,
 123                         NULL, &NewNode);
 124         if (ACPI_FAILURE (Status))
```

```
125         if (ACPI_FAILURE (Status) || (!NewNode)) /* Must be on same line for cod
125         {
126             ACPI_EXCEPTION ((AE_INFO, Status,
127                 "Could not create predefined name %s",
128                 InitVal->Name));
129             continue;
130         }

132         /*
133          * Name entered successfully. If entry in PreDefinedNames[] specifies
134          * an initial value, create the initial value.
135          */
136         if (InitVal->Val)
137         {
138             Status = AcpiOsPredefinedOverride (InitVal, &Val);
139             if (ACPI_FAILURE (Status))
140             {
141                 ACPI_ERROR ((AE_INFO,
142                     "Could not override predefined %s",
143                     InitVal->Name));
144             }

146             if (!Val)
147             {
148                 Val = InitVal->Val;
149             }

151             /*
152              * Entry requests an initial value, allocate a
153              * descriptor for it.
154              */
155             ObjDesc = AcpiUtCreateInternalObject (InitVal->Type);
156             if (!ObjDesc)
157             {
158                 Status = AE_NO_MEMORY;
159                 goto UnlockAndExit;
160             }

162             /*
163              * Convert value string from table entry to
164              * internal representation. Only types actually
165              * used for initial values are implemented here.
166              */
167             switch (InitVal->Type)
168             {
169             case ACPI_TYPE_METHOD:

171                 ObjDesc->Method.ParamCount = (UINT8) ACPI_TO_INTEGER (Val);
172                 ObjDesc->Common.Flags |= AOPOBJ_DATA_VALID;

174 #if defined (ACPI_ASL_COMPILER)

176                 /* Save the parameter count for the iASL compiler */

178                 NewNode->Value = ObjDesc->Method.ParamCount;
179 #else
180                 /* Mark this as a very SPECIAL method */

182                 ObjDesc->Method.InfoFlags = ACPI_METHOD_INTERNAL_ONLY;
183                 ObjDesc->Method.Dispatch.Implementation = AcpiUtOsiImplementatio
184 #endif
185                 break;

187             case ACPI_TYPE_INTEGER:

189                 ObjDesc->Integer.Value = ACPI_TO_INTEGER (Val);
```



```
190                 break;


192             case ACPI_TYPE_STRING:

194                 /* Build an object around the static string */

196                 ObjDesc->String.Length = (UINT32) ACPI_STRLEN (Val);
197                 ObjDesc->String.Pointer = Val;
198                 ObjDesc->Common.Flags |= AOPOBJ_STATIC_POINTER;
199                 break;


201             case ACPI_TYPE_MUTEX:

203                 ObjDesc->Mutex.Node = NewNode;
204                 ObjDesc->Mutex.SyncLevel = (UINT8) (ACPI_TO_INTEGER (Val) - 1);

206                 /* Create a mutex */

208                 Status = AcpiOsCreateMutex (&ObjDesc->Mutex.OsMutex);
209                 if (ACPI_FAILURE (Status))
210                 {
211                     AcpiUtRemoveReference (ObjDesc);
212                     goto UnlockAndExit;
213                 }

215                 /* Special case for ACPI Global Lock */

217                 if (ACPI_STRCMP (InitVal->Name, "_GL_") == 0)
218                 {
219                     AcpiGbl_GlobalLockMutex = ObjDesc;

221                     /* Create additional counting semaphore for global lock */

223                     Status = AcpiOsCreateSemaphore (
224                                 1, 0, &AcpiGbl_GlobalLockSemaphore);
225                     if (ACPI_FAILURE (Status))
226                     {
227                         AcpiUtRemoveReference (ObjDesc);
228                         goto UnlockAndExit;
229                     }
230                 }
231                 break;


233             default:

235                 ACPI_ERROR ((AE_INFO, "Unsupported initial type value 0x%X",
236                     InitVal->Type));
237                 AcpiUtRemoveReference (ObjDesc);
238                 ObjDesc = NULL;
239                 continue;
240             }

242             /* Store pointer to value descriptor in the Node */

244             Status = AcpiNsAttachObject (NewNode, ObjDesc,
245                         ObjDesc->Common.Type);

247             /* Remove local reference to the object */

249             AcpiUtRemoveReference (ObjDesc);
250         }
251     }
```

```
254 UnlockAndExit:
255     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

257     /* Save a handle to "_GPE", it is always present */

259     if (ACPI_SUCCESS (Status))
260     {
261         Status = AcpiNsGetNode (NULL, "\\_GPE", ACPI_NS_NO_UPSEARCH,
262                     &AcpiGbl_FadtGpeDevice);
263     }

265     return_ACPI_STATUS (Status);
266 }


269 /*******************************************************************************
270  *
271  * FUNCTION:    AcpiNsLookup
272  *
273  * PARAMETERS:  ScopeInfo       - Current scope info block
274  *              Pathname        - Search pathname, in internal format
275  *                                (as represented in the AML stream)
276  *              Type            - Type associated with name
277  *              InterpreterMode - IMODE_LOAD_PASS2 => add name if not found
278  *              Flags           - Flags describing the search restrictions
279  *              WalkState       - Current state of the walk
280  *              ReturnNode      - Where the Node is placed (if found
281  *                                or created successfully)
282  *
283  * RETURN:      Status
284  *
285  * DESCRIPTION: Find or enter the passed name in the name space.
286  *              Log an error if name not found in Exec mode.
287  *
288  * MUTEX:       Assumes namespace is locked.
289  *
290  ******************************************************************************/

292 ACPI_STATUS
293 AcpiNsLookup (
294     ACPI_GENERIC_STATE      *ScopeInfo,
295     char                    *Pathname,
296     ACPI_OBJECT_TYPE        Type,
297     ACPI_INTERPRETER_MODE   InterpreterMode,
298     UINT32                  Flags,
299     ACPI_WALK_STATE         *WalkState,
300     ACPI_NAMESPACE_NODE     **ReturnNode)
301 {
302     ACPI_STATUS             Status;
303     char                    *Path = Pathname;
304     ACPI_NAMESPACE_NODE     *PrefixNode;
305     ACPI_NAMESPACE_NODE     *CurrentNode = NULL;
306     ACPI_NAMESPACE_NODE     *ThisNode = NULL;
307     UINT32                  NumSegments;
308     UINT32                  NumCarats;
309     ACPI_NAME               SimpleName;
310     ACPI_OBJECT_TYPE        TypeToCheckFor;
311     ACPI_OBJECT_TYPE        ThisSearchType;
312     UINT32                  SearchParentFlag = ACPI_NS_SEARCH_PARENT;
313     UINT32                  LocalFlags;


316     ACPI_FUNCTION_TRACE (NsLookup);
```



```
319     if (!ReturnNode)
320     {
321         return_ACPI_STATUS (AE_BAD_PARAMETER);
322     }

324     LocalFlags = Flags & ~(ACPI_NS_ERROR_IF_FOUND | ACPI_NS_SEARCH_PARENT);
325     *ReturnNode = ACPI_ENTRY_NOT_FOUND;
326     AcpiGbl_NsLookupCount++;

328     if (!AcpiGbl_RootNode)
329     {
330         return_ACPI_STATUS (AE_NO_NAMESPACE);
331     }

333     /* Get the prefix scope. A null scope means use the root scope */

335     if ((!ScopeInfo) ||
336         (!ScopeInfo->Scope.Node))
337     {
338         ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
339             "Null scope prefix, using root node (%p)\n",
340             AcpiGbl_RootNode));

342         PrefixNode = AcpiGbl_RootNode;
343     }
344     else
345     {
346         PrefixNode = ScopeInfo->Scope.Node;
347         if (ACPI_GET_DESCRIPTOR_TYPE (PrefixNode) != ACPI_DESC_TYPE_NAMED)
348         {
349             ACPI_ERROR ((AE_INFO, "%p is not a namespace node [%s]",
350                 PrefixNode, AcpiUtGetDescriptorName (PrefixNode)));
351             return_ACPI_STATUS (AE_AML_INTERNAL);
352         }

354         if (!(Flags & ACPI_NS_PREFIX_IS_SCOPE))
355         {
356             /*
357              * This node might not be a actual "scope" node (such as a
358              * Device/Method, etc.)  It could be a Package or other object
359              * node. Backup up the tree to find the containing scope node.
360              */
361             while (!AcpiNsOpensScope (PrefixNode->Type) &&
362                     PrefixNode->Type != ACPI_TYPE_ANY)
363             {
364                 PrefixNode = PrefixNode->Parent;
365             }
366         }
367     }

369     /* Save type. TBD: may be no longer necessary */

371     TypeToCheckFor = Type;

373     /*
374      * Begin examination of the actual pathname
375      */
376     if (!Pathname)
377     {
378         /* A Null NamePath is allowed and refers to the root */

380         NumSegments = 0;
381         ThisNode = AcpiGbl_RootNode;
382         Path = "";

384         ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
```

```
385             "Null Pathname (Zero segments), Flags=%X\n", Flags));
386     }
387     else
388     {
389         /*
390          * Name pointer is valid (and must be in internal name format)
391          *
392          * Check for scope prefixes:
393          *
394          * As represented in the AML stream, a namepath consists of an
395          * optional scope prefix followed by a name segment part.
396          *
397          * If present, the scope prefix is either a Root Prefix (in
398          * which case the name is fully qualified), or one or more
399          * Parent Prefixes (in which case the name's scope is relative
400          * to the current scope).
401          */
402         if (*Path == (UINT8) AML_ROOT_PREFIX)
403         {
404             /* Pathname is fully qualified, start from the root */

406             ThisNode = AcpiGbl_RootNode;
407             SearchParentFlag = ACPI_NS_NO_UPSEARCH;

409             /* Point to name segment part */

411             Path++;

413             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
414                 "Path is absolute from root [%p]\n", ThisNode));
415         }
416         else
417         {
418             /* Pathname is relative to current scope, start there */

420             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
421                 "Searching relative to prefix scope [%4.4s] (%p)\n",
422                 AcpiUtGetNodeName (PrefixNode), PrefixNode));

424             /*
425              * Handle multiple Parent Prefixes (carat) by just getting
426              * the parent node for each prefix instance.
427              */
428             ThisNode = PrefixNode;
429             NumCarats = 0;
430             while (*Path == (UINT8) AML_PARENT_PREFIX)
431             {
432                 /* Name is fully qualified, no search rules apply */

434                 SearchParentFlag = ACPI_NS_NO_UPSEARCH;

436                 /*
437                  * Point past this prefix to the name segment
438                  * part or the next Parent Prefix
439                  */
440                 Path++;

442                 /* Backup to the parent node */

444                 NumCarats++;
445                 ThisNode = ThisNode->Parent;
446                 if (!ThisNode)
447                 {
448                     /* Current scope has no parent scope */

450                     ACPI_ERROR ((AE_INFO,
```



```
451                         "%s: Path has too many parent prefixes (^) "
452                         "- reached beyond root node", Pathname));
453                         "ACPI path has too many parent prefixes (^) "
454                         "- reached beyond root node"));
453                     return_ACPI_STATUS (AE_NOT_FOUND);
454                 }
455             }

457             if (SearchParentFlag == ACPI_NS_NO_UPSEARCH)
458             {
459                 ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
460                     "Search scope is [%4.4s], path has %u carat(s)\n",
461                     AcpiUtGetNodeName (ThisNode), NumCarats));
462             }
463         }

465         /*
466          * Determine the number of ACPI name segments in this pathname.
467          *
468          * The segment part consists of either:
469          *  - A Null name segment (0)
470          *  - A DualNamePrefix followed by two 4-byte name segments
471          *  - A MultiNamePrefix followed by a byte indicating the
472          *      number of segments and the segments themselves.
473          *  - A single 4-byte name segment
474          *
475          * Examine the name prefix opcode, if any, to determine the number of
476          * segments.
477          */
478         switch (*Path)
479         {
480         case 0:
481             /*
482              * Null name after a root or parent prefixes. We already
483              * have the correct target node and there are no name segments.
484              */
485             NumSegments  = 0;
486             Type = ThisNode->Type;

488             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
489                 "Prefix-only Pathname (Zero name segments), Flags=%X\n",
490                 Flags));
491             break;

493         case AML_DUAL_NAME_PREFIX:

495             /* More than one NameSeg, search rules do not apply */

497             SearchParentFlag = ACPI_NS_NO_UPSEARCH;

499             /* Two segments, point to first name segment */

501             NumSegments = 2;
502             Path++;

504             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
505                 "Dual Pathname (2 segments, Flags=%X)\n", Flags));
506             break;

508         case AML_MULTI_NAME_PREFIX_OP:

510             /* More than one NameSeg, search rules do not apply */

512             SearchParentFlag = ACPI_NS_NO_UPSEARCH;

514             /* Extract segment count, point to first name segment */
```

```
516             Path++;
517             NumSegments = (UINT32) (UINT8) *Path;
518             Path++;

520             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
521                 "Multi Pathname (%u Segments, Flags=%X)\n",
522                 NumSegments, Flags));
523             break;

525         default:
526             /*
527              * Not a Null name, no Dual or Multi prefix, hence there is
528              * only one name segment and Pathname is already pointing to it.
529              */
530             NumSegments = 1;

532             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
533                 "Simple Pathname (1 segment, Flags=%X)\n", Flags));
534             break;
535         }

537         ACPI_DEBUG_EXEC (AcpiNsPrintPathname (NumSegments, Path));
538     }


541     /*
542      * Search namespace for each segment of the name. Loop through and
543      * verify (or add to the namespace) each name segment.
544      *
545      * The object type is significant only at the last name
546      * segment. (We don't care about the types along the path, only
547      * the type of the final target object.)
548      */
549     ThisSearchType = ACPI_TYPE_ANY;
550     CurrentNode = ThisNode;
551     while (NumSegments && CurrentNode)
552     {
553         NumSegments--;
554         if (!NumSegments)
555         {
556             /* This is the last segment, enable typechecking */

558             ThisSearchType = Type;

560             /*
561              * Only allow automatic parent search (search rules) if the caller
562              * requested it AND we have a single, non-fully-qualified NameSeg
563              */
564             if ((SearchParentFlag != ACPI_NS_NO_UPSEARCH) &&
565                 (Flags & ACPI_NS_SEARCH_PARENT))
566             {
567                 LocalFlags |= ACPI_NS_SEARCH_PARENT;
568             }

570             /* Set error flag according to caller */

572             if (Flags & ACPI_NS_ERROR_IF_FOUND)
573             {
574                 LocalFlags |= ACPI_NS_ERROR_IF_FOUND;
575             }
576         }

578         /* Extract one ACPI name from the front of the pathname */

580         ACPI_MOVE_32_TO_32 (&SimpleName, Path);
```



```
582         /* Try to find the single (4 character) ACPI name */

584         Status = AcpiNsSearchAndEnter (SimpleName, WalkState, CurrentNode,
585                     InterpreterMode, ThisSearchType, LocalFlags, &ThisNode);
586         if (ACPI_FAILURE (Status))
587         {
588             if (Status == AE_NOT_FOUND)
589             {
590                 /* Name not found in ACPI namespace */

592                 ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
593                     "Name [%4.4s] not found in scope [%4.4s] %p\n",
594                     (char *) &SimpleName, (char *) &CurrentNode->Name,
595                     CurrentNode));
596             }

598             *ReturnNode = ThisNode;
599             return_ACPI_STATUS (Status);
600         }

602         /* More segments to follow? */

604         if (NumSegments > 0)
605         {
606             /*
607              * If we have an alias to an object that opens a scope (such as a
608              * device or processor), we need to dereference the alias here so
609              * that we can access any children of the original node (via the
610              * remaining segments).
611              */
612             if (ThisNode->Type == ACPI_TYPE_LOCAL_ALIAS)
613             {
614                 if (!ThisNode->Object)
615                 {
616                     return_ACPI_STATUS (AE_NOT_EXIST);
617                 }

619                 if (AcpiNsOpensScope (((ACPI_NAMESPACE_NODE *)
620                         ThisNode->Object)->Type))
621                 {
622                     ThisNode = (ACPI_NAMESPACE_NODE *) ThisNode->Object;
623                 }
624             }
625         }

627         /* Special handling for the last segment (NumSegments == 0) */

629         else
630         {
631             /*
632              * Sanity typecheck of the target object:
633              *
634              * If 1) This is the last segment (NumSegments == 0)
635              *    2) And we are looking for a specific type
636              *       (Not checking for TYPE_ANY)
637              *    3) Which is not an alias
638              *    4) Which is not a local type (TYPE_SCOPE)
639              *    5) And the type of target object is known (not TYPE_ANY)
640              *    6) And target object does not match what we are looking for
641              *
642              * Then we have a type mismatch. Just warn and ignore it.
643              */
644             if ((TypeToCheckFor != ACPI_TYPE_ANY)                   &&
645                 (TypeToCheckFor != ACPI_TYPE_LOCAL_ALIAS)           &&
646                 (TypeToCheckFor != ACPI_TYPE_LOCAL_METHOD_ALIAS)    &&
```

```
647                 (TypeToCheckFor != ACPI_TYPE_LOCAL_SCOPE)           &&
648                 (ThisNode->Type != ACPI_TYPE_ANY)                   &&
649                 (ThisNode->Type != TypeToCheckFor))
650             {
651                 /* Complain about a type mismatch */

653                 ACPI_WARNING ((AE_INFO,
654                     "NsLookup: Type mismatch on %4.4s (%s), searching for (%s)",
655                     ACPI_CAST_PTR (char, &SimpleName),
656                     AcpiUtGetTypeName (ThisNode->Type),
657                     AcpiUtGetTypeName (TypeToCheckFor)));
658             }

660             /*
661              * If this is the last name segment and we are not looking for a
662              * specific type, but the type of found object is known, use that
663              * type to (later) see if it opens a scope.
664              */
665             if (Type == ACPI_TYPE_ANY)
666             {
667                 Type = ThisNode->Type;
668             }
669         }

671         /* Point to next name segment and make this node current */

673         Path += ACPI_NAME_SIZE;
674         CurrentNode = ThisNode;
675     }

677     /* Always check if we need to open a new scope */

679     if (!(Flags & ACPI_NS_DONT_OPEN_SCOPE) && (WalkState))
680     {
681         /*
682          * If entry is a type which opens a scope, push the new scope on the
683          * scope stack.
684          */
685         if (AcpiNsOpensScope (Type))
686         {
687             Status = AcpiDsScopeStackPush (ThisNode, Type, WalkState);
688             if (ACPI_FAILURE (Status))
689             {
690                 return_ACPI_STATUS (Status);
691             }
692         }
693     }

695     *ReturnNode = ThisNode;
696     return_ACPI_STATUS (AE_OK);
697 }
```

```
**********************************************************
   16136 Thu Dec 26 13:49:16 2013
new/usr/src/common/acpica/components/namespace/nsalloc.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsalloc - Namespace allocation and deletion utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSALLOC_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsalloc")


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsCreateNode
```



```
 *
 * PARAMETERS:  Name            - Name of the new node (4 char ACPI name)
 *
 * RETURN:      New namespace node (Null on failure)
 *
 * DESCRIPTION: Create a namespace node
 *
 ******************************************************************************/

ACPI_NAMESPACE_NODE *
AcpiNsCreateNode (
    UINT32                  Name)
{
    ACPI_NAMESPACE_NODE     *Node;
#ifdef ACPI_DBG_TRACK_ALLOCATIONS
    UINT32                  Temp;
#endif


    ACPI_FUNCTION_TRACE (NsCreateNode);


    Node = AcpiOsAcquireObject (AcpiGbl_NamespaceCache);
    if (!Node)
    {
        return_PTR (NULL);
    }

    ACPI_MEM_TRACKING (AcpiGbl_NsNodeList->TotalAllocated++);

#ifdef ACPI_DBG_TRACK_ALLOCATIONS
        Temp = AcpiGbl_NsNodeList->TotalAllocated -
                AcpiGbl_NsNodeList->TotalFreed;
        if (Temp > AcpiGbl_NsNodeList->MaxOccupied)
        {
            AcpiGbl_NsNodeList->MaxOccupied = Temp;
        }
#endif

    Node->Name.Integer = Name;
    ACPI_SET_DESCRIPTOR_TYPE (Node, ACPI_DESC_TYPE_NAMED);
    return_PTR (Node);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsDeleteNode
 *
 * PARAMETERS:  Node            - Node to be deleted
 *
 * RETURN:      None
 *
 * DESCRIPTION: Delete a namespace node. All node deletions must come through
 *              here. Detaches any attached objects, including any attached
 *              data. If a handler is associated with attached data, it is
 *              invoked before the node is deleted.
 *
 ******************************************************************************/

void
AcpiNsDeleteNode (
    ACPI_NAMESPACE_NODE     *Node)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     *NextDesc;
```

```
127     ACPI_FUNCTION_NAME (NsDeleteNode);


130     /* Detach an object if there is one */

132     AcpiNsDetachObject (Node);

134     /*
135      * Delete an attached data object list if present (objects that were
136      * attached via AcpiAttachData). Note: After any normal object is
137      * detached above, the only possible remaining object(s) are data
138      * objects, in a linked list.
134      * Delete an attached data object if present (an object that was created
135      * and attached via AcpiAttachData). Note: After any normal object is
136      * detached above, the only possible remaining object is a data object.
139      */
140     ObjDesc = Node->Object;
141     while (ObjDesc &&
139     if (ObjDesc &&
142         (ObjDesc->Common.Type == ACPI_TYPE_LOCAL_DATA))
143     {
144         /* Invoke the attached data deletion handler if present */

146         if (ObjDesc->Data.Handler)
147         {
148             ObjDesc->Data.Handler (Node, ObjDesc->Data.Pointer);
149         }

151         NextDesc = ObjDesc->Common.NextObject;
152         AcpiUtRemoveReference (ObjDesc);
153         ObjDesc = NextDesc;
154     }

156     /* Special case for the statically allocated root node */

158     if (Node == AcpiGbl_RootNode)
159     {
160         return;
161     }

163     /* Now we can delete the node */

165     (void) AcpiOsReleaseObject (AcpiGbl_NamespaceCache, Node);

167     ACPI_MEM_TRACKING (AcpiGbl_NsNodeList->TotalFreed++);
168     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Node %p, Remaining %X\n",
169         Node, AcpiGbl_CurrentNodeCount));
170 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   10923 Thu Dec 26 13:49:16 2013
new/usr/src/common/acpica/components/namespace/nsarguments.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsarguments - Validation of args for ACPI predefined methods
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acpredef.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsarguments")


/******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckArgumentTypes
 *
 * PARAMETERS:  Info            - Method execution information block
 *
 * RETURN:      None
 *
```



```
 * DESCRIPTION: Check the incoming argument count and all argument types
 *              against the argument type list for a predefined name.
 *
 *****************************************************************************/

void
AcpiNsCheckArgumentTypes (
    ACPI_EVALUATE_INFO          *Info)
{
    UINT16                      ArgTypeList;
    UINT8                       ArgCount;
    UINT8                       ArgType;
    UINT8                       UserArgType;
    UINT32                      i;


    /* If not a predefined name, cannot typecheck args */

    if (!Info->Predefined)
    {
        return;
    }

    ArgTypeList = Info->Predefined->Info.ArgumentList;
    ArgCount = METHOD_GET_ARG_COUNT (ArgTypeList);

    /* Typecheck all arguments */

    for (i = 0; ((i < ArgCount) && (i < Info->ParamCount)); i++)
    {
        ArgType = METHOD_GET_NEXT_TYPE (ArgTypeList);
        UserArgType = Info->Parameters[i]->Common.Type;

        if (UserArgType != ArgType)
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, ACPI_WARN_ALWAYS
                "Argument #%u type mismatch - "
                "Found [%s], ACPI requires [%s]", (i + 1),
                AcpiUtGetTypeName (UserArgType),
                AcpiUtGetTypeName (ArgType)));
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckAcpiCompliance
 *
 * PARAMETERS:  Pathname        - Full pathname to the node (for error msgs)
 *              Node            - Namespace node for the method/object
 *              Predefined      - Pointer to entry in predefined name table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Check that the declared parameter count (in ASL/AML) for a
 *              predefined name is what is expected (matches what is defined in
 *              the ACPI specification for this predefined name.)
 *
 *****************************************************************************/

void
AcpiNsCheckAcpiCompliance (
    char                        *Pathname,
    ACPI_NAMESPACE_NODE         *Node,
    const ACPI_PREDEFINED_INFO  *Predefined)
```

```
{
    UINT32                  AmlParamCount;
    UINT32                  RequiredParamCount;


    if (!Predefined)
    {
        return;
    }

    /* Get the ACPI-required arg count from the predefined info table */

    RequiredParamCount = METHOD_GET_ARG_COUNT (Predefined->Info.ArgumentList);

    /*
     * If this object is not a control method, we can check if the ACPI
     * spec requires that it be a method.
     */
    if (Node->Type != ACPI_TYPE_METHOD)
    {
        if (RequiredParamCount > 0)
        {
            /* Object requires args, must be implemented as a method */

            ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
                "Object (%s) must be a control method with %u arguments",
                AcpiUtGetTypeName (Node->Type), RequiredParamCount));
        }
        else if (!RequiredParamCount && !Predefined->Info.ExpectedBtypes)
        {
            /* Object requires no args and no return value, must be a method */

            ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
                "Object (%s) must be a control method "
                "with no arguments and no return value",
                AcpiUtGetTypeName (Node->Type)));
        }

        return;
    }

    /*
     * This is a control method.
     * Check that the ASL/AML-defined parameter count for this method
     * matches the ACPI-required parameter count
     *
     * Some methods are allowed to have a "minimum" number of args (_SCP)
     * because their definition in ACPI has changed over time.
     *
     * Note: These are BIOS errors in the declaration of the object
     */
    AmlParamCount = Node->Object->Method.ParamCount;

    if (AmlParamCount < RequiredParamCount)
    {
        ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
            "Insufficient arguments - "
            "ASL declared %u, ACPI requires %u",
            AmlParamCount, RequiredParamCount));
    }
    else if ((AmlParamCount > RequiredParamCount) &&
        !(Predefined->Info.ArgumentList & ARG_COUNT_IS_MINIMUM))
    {
        ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
            "Excess arguments - "
            "ASL declared %u, ACPI requires %u",
```



```
            AmlParamCount, RequiredParamCount));
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckArgumentCount
 *
 * PARAMETERS:  Pathname        - Full pathname to the node (for error msgs)
 *              Node            - Namespace node for the method/object
 *              UserParamCount  - Number of args passed in by the caller
 *              Predefined      - Pointer to entry in predefined name table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Check that incoming argument count matches the declared
 *              parameter count (in the ASL/AML) for an object.
 *
 ******************************************************************************/

void
AcpiNsCheckArgumentCount (
    char                        *Pathname,
    ACPI_NAMESPACE_NODE         *Node,
    UINT32                      UserParamCount,
    const ACPI_PREDEFINED_INFO  *Predefined)
{
    UINT32                      AmlParamCount;
    UINT32                      RequiredParamCount;


    if (!Predefined)
    {
        /*
         * Not a predefined name. Check the incoming user argument count
         * against the count that is specified in the method/object.
         */
        if (Node->Type != ACPI_TYPE_METHOD)
        {
            if (UserParamCount)
            {
                ACPI_INFO_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
                    "%u arguments were passed to a non-method ACPI object (%s)",
                    UserParamCount, AcpiUtGetTypeName (Node->Type)));
            }

            return;
        }

        /*
         * This is a control method. Check the parameter count.
         * We can only check the incoming argument count against the
         * argument count declared for the method in the ASL/AML.
         *
         * Emit a message if too few or too many arguments have been passed
         * by the caller.
         *
         * Note: Too many arguments will not cause the method to
         * fail. However, the method will fail if there are too few
         * arguments and the method attempts to use one of the missing ones.
         */
        AmlParamCount = Node->Object->Method.ParamCount;

        if (UserParamCount < AmlParamCount)
        {
```

```
260             ACPI_WARN_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
261                 "Insufficient arguments - "
262                 "Caller passed %u, method requires %u",
263                 UserParamCount, AmlParamCount));
264         }
265         else if (UserParamCount > AmlParamCount)
266         {
267             ACPI_INFO_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
268                 "Excess arguments - "
269                 "Caller passed %u, method requires %u",
270                 UserParamCount, AmlParamCount));
271         }

273         return;
274     }

276     /*
277      * This is a predefined name. Validate the user-supplied parameter
278      * count against the ACPI specification. We don't validate against
279      * the method itself because what is important here is that the
280      * caller is in conformance with the spec. (The arg count for the
281      * method was checked against the ACPI spec earlier.)
282      *
283      * Some methods are allowed to have a "minimum" number of args (_SCP)
284      * because their definition in ACPI has changed over time.
285      */
286     RequiredParamCount = METHOD_GET_ARG_COUNT (Predefined->Info.ArgumentList);

288     if (UserParamCount < RequiredParamCount)
289     {
290         ACPI_WARN_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
291             "Insufficient arguments - "
292             "Caller passed %u, ACPI requires %u",
293             UserParamCount, RequiredParamCount));
294     }
295     else if ((UserParamCount > RequiredParamCount) &&
296         !(Predefined->Info.ArgumentList & ARG_COUNT_IS_MINIMUM))
297     {
298         ACPI_INFO_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
299             "Excess arguments - "
300             "Caller passed %u, ACPI requires %u",
301             UserParamCount, RequiredParamCount));
302     }
303 }
```

```
**********************************************************
   14083 Thu Dec 26 13:49:16 2013
new/usr/src/common/acpica/components/namespace/nsconvert.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsconvert - Object conversions for objects returned by
 *                          predefined methods
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSCONVERT_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acinterp.h"
#include "acpredef.h"
#include "amlresrc.h"

#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsconvert")


/******************************************************************************
 *
 * FUNCTION:    AcpiNsConvertToInteger
 *
```



```
 * PARAMETERS:  OriginalObject      - Object to be converted
 *              ReturnObject        - Where the new converted object is returned
 *
 * RETURN:      Status. AE_OK if conversion was successful.
 *
 * DESCRIPTION: Attempt to convert a String/Buffer object to an Integer.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsConvertToInteger (
    ACPI_OPERAND_OBJECT     *OriginalObject,
    ACPI_OPERAND_OBJECT     **ReturnObject)
{
    ACPI_OPERAND_OBJECT     *NewObject;
    ACPI_STATUS             Status;
    UINT64                  Value = 0;
    UINT32                  i;


    switch (OriginalObject->Common.Type)
    {
    case ACPI_TYPE_STRING:

        /* String-to-Integer conversion */

        Status = AcpiUtStrtoul64 (OriginalObject->String.Pointer,
                    ACPI_ANY_BASE, &Value);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }
        break;

    case ACPI_TYPE_BUFFER:

        /* Buffer-to-Integer conversion. Max buffer size is 64 bits. */

        if (OriginalObject->Buffer.Length > 8)
        {
            return (AE_AML_OPERAND_TYPE);
        }

        /* Extract each buffer byte to create the integer */

        for (i = 0; i < OriginalObject->Buffer.Length; i++)
        {
            Value |= ((UINT64) OriginalObject->Buffer.Pointer[i] << (i * 8));
        }
        break;

    default:

        return (AE_AML_OPERAND_TYPE);
    }

    NewObject = AcpiUtCreateIntegerObject (Value);
    if (!NewObject)
    {
        return (AE_NO_MEMORY);
    }

    *ReturnObject = NewObject;
    return (AE_OK);
}
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiNsConvertToString
 *
 * PARAMETERS:  OriginalObject      - Object to be converted
 *              ReturnObject        - Where the new converted object is returned
 *
 * RETURN:      Status. AE_OK if conversion was successful.
 *
 * DESCRIPTION: Attempt to convert a Integer/Buffer object to a String.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiNsConvertToString (
    ACPI_OPERAND_OBJECT     *OriginalObject,
    ACPI_OPERAND_OBJECT     **ReturnObject)
{
    ACPI_OPERAND_OBJECT     *NewObject;
    ACPI_SIZE               Length;
    ACPI_STATUS             Status;


    switch (OriginalObject->Common.Type)
    {
    case ACPI_TYPE_INTEGER:
        /*
         * Integer-to-String conversion. Commonly, convert
         * an integer of value 0 to a NULL string. The last element of
         * _BIF and _BIX packages occasionally need this fix.
         */
        if (OriginalObject->Integer.Value == 0)
        {
            /* Allocate a new NULL string object */

            NewObject = AcpiUtCreateStringObject (0);
            if (!NewObject)
            {
                return (AE_NO_MEMORY);
            }
        }
        else
        {
            Status = AcpiExConvertToString (OriginalObject, &NewObject,
                        ACPI_IMPLICIT_CONVERT_HEX);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
        }
        break;

    case ACPI_TYPE_BUFFER:
        /*
         * Buffer-to-String conversion. Use a ToString
         * conversion, no transform performed on the buffer data. The best
         * example of this is the _BIF method, where the string data from
         * the battery is often (incorrectly) returned as buffer object(s).
         */
        Length = 0;
        while ((Length < OriginalObject->Buffer.Length) &&
               (OriginalObject->Buffer.Pointer[Length]))
        {
            Length++;
        }
```



```
        /* Allocate a new string object */

        NewObject = AcpiUtCreateStringObject (Length);
        if (!NewObject)
        {
            return (AE_NO_MEMORY);
        }

        /*
         * Copy the raw buffer data with no transform. String is already NULL
         * terminated at Length+1.
         */
        ACPI_MEMCPY (NewObject->String.Pointer,
            OriginalObject->Buffer.Pointer, Length);
        break;

    default:

        return (AE_AML_OPERAND_TYPE);
    }

    *ReturnObject = NewObject;
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsConvertToBuffer
 *
 * PARAMETERS:  OriginalObject      - Object to be converted
 *              ReturnObject        - Where the new converted object is returned
 *
 * RETURN:      Status. AE_OK if conversion was successful.
 *
 * DESCRIPTION: Attempt to convert a Integer/String/Package object to a Buffer.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiNsConvertToBuffer (
    ACPI_OPERAND_OBJECT     *OriginalObject,
    ACPI_OPERAND_OBJECT     **ReturnObject)
{
    ACPI_OPERAND_OBJECT     *NewObject;
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     **Elements;
    UINT32                  *DwordBuffer;
    UINT32                  Count;
    UINT32                  i;


    switch (OriginalObject->Common.Type)
    {
    case ACPI_TYPE_INTEGER:
        /*
         * Integer-to-Buffer conversion.
         * Convert the Integer to a packed-byte buffer. _MAT and other
         * objects need this sometimes, if a read has been performed on a
         * Field object that is less than or equal to the global integer
         * size (32 or 64 bits).
         */
        Status = AcpiExConvertToBuffer (OriginalObject, &NewObject);
        if (ACPI_FAILURE (Status))
        {
```

```
260             return (Status);
261         }
262         break;

264     case ACPI_TYPE_STRING:

266         /* String-to-Buffer conversion. Simple data copy */

268         NewObject = AcpiUtCreateBufferObject (OriginalObject->String.Length);
269         if (!NewObject)
270         {
271             return (AE_NO_MEMORY);
272         }

274         ACPI_MEMCPY (NewObject->Buffer.Pointer,
275             OriginalObject->String.Pointer, OriginalObject->String.Length);
276         break;

278     case ACPI_TYPE_PACKAGE:
279         /*
280          * This case is often seen for predefined names that must return a
281          * Buffer object with multiple DWORD integers within. For example,
282          * _FDE and _GTM. The Package can be converted to a Buffer.
283          */

285         /* All elements of the Package must be integers */

287         Elements = OriginalObject->Package.Elements;
288         Count = OriginalObject->Package.Count;

290         for (i = 0; i < Count; i++)
291         {
292             if ((!*Elements) ||
293                 ((*Elements)->Common.Type != ACPI_TYPE_INTEGER))
294             {
295                 return (AE_AML_OPERAND_TYPE);
296             }
297             Elements++;
298         }

300         /* Create the new buffer object to replace the Package */

302         NewObject = AcpiUtCreateBufferObject (ACPI_MUL_4 (Count));
303         if (!NewObject)
304         {
305             return (AE_NO_MEMORY);
306         }

308         /* Copy the package elements (integers) to the buffer as DWORDs */

310         Elements = OriginalObject->Package.Elements;
311         DwordBuffer = ACPI_CAST_PTR (UINT32, NewObject->Buffer.Pointer);

313         for (i = 0; i < Count; i++)
314         {
315             *DwordBuffer = (UINT32) (*Elements)->Integer.Value;
316             DwordBuffer++;
317             Elements++;
318         }
319         break;

321     default:

323         return (AE_AML_OPERAND_TYPE);
324     }
```



```
326     *ReturnObject = NewObject;
327     return (AE_OK);
328 }


331 /*******************************************************************************
332  *
333  * FUNCTION:    AcpiNsConvertToUnicode
334  *
335  * PARAMETERS:  OriginalObject      - ASCII String Object to be converted
336  *              ReturnObject        - Where the new converted object is returned
337  *
338  * RETURN:      Status. AE_OK if conversion was successful.
339  *
340  * DESCRIPTION: Attempt to convert a String object to a Unicode string Buffer.
341  *
342  ******************************************************************************/

344 ACPI_STATUS
345 AcpiNsConvertToUnicode (
346     ACPI_OPERAND_OBJECT     *OriginalObject,
347     ACPI_OPERAND_OBJECT     **ReturnObject)
348 {
349     ACPI_OPERAND_OBJECT     *NewObject;
350     char                    *AsciiString;
351     UINT16                  *UnicodeBuffer;
352     UINT32                  UnicodeLength;
353     UINT32                  i;


356     if (!OriginalObject)
357     {
358         return (AE_OK);
359     }

361     /* If a Buffer was returned, it must be at least two bytes long */

363     if (OriginalObject->Common.Type == ACPI_TYPE_BUFFER)
364     {
365         if (OriginalObject->Buffer.Length < 2)
366         {
367             return (AE_AML_OPERAND_VALUE);
368         }

370         *ReturnObject = NULL;
371         return (AE_OK);
372     }

374     /*
375      * The original object is an ASCII string. Convert this string to
376      * a unicode buffer.
377      */
378     AsciiString = OriginalObject->String.Pointer;
379     UnicodeLength = (OriginalObject->String.Length * 2) + 2;

381     /* Create a new buffer object for the Unicode data */

383     NewObject = AcpiUtCreateBufferObject (UnicodeLength);
384     if (!NewObject)
385     {
386         return (AE_NO_MEMORY);
387     }

389     UnicodeBuffer = ACPI_CAST_PTR (UINT16, NewObject->Buffer.Pointer);

391     /* Convert ASCII to Unicode */
```

```
393     for (i = 0; i < OriginalObject->String.Length; i++)
394     {
395         UnicodeBuffer[i] = (UINT16) AsciiString[i];
396     }

398     *ReturnObject = NewObject;
399     return (AE_OK);
400 }


403 /*******************************************************************************
404  *
405  * FUNCTION:    AcpiNsConvertToResource
406  *
407  * PARAMETERS:  OriginalObject      - Object to be converted
408  *              ReturnObject        - Where the new converted object is returned
409  *
410  * RETURN:      Status. AE_OK if conversion was successful
411  *
412  * DESCRIPTION: Attempt to convert a Integer object to a ResourceTemplate
413  *              Buffer.
414  *
415  ******************************************************************************/

417 ACPI_STATUS
418 AcpiNsConvertToResource (
419     ACPI_OPERAND_OBJECT     *OriginalObject,
420     ACPI_OPERAND_OBJECT     **ReturnObject)
421 {
422     ACPI_OPERAND_OBJECT     *NewObject;
423     UINT8                   *Buffer;


426     /*
427      * We can fix the following cases for an expected resource template:
428      * 1. No return value (interpreter slack mode is disabled)
429      * 2. A "Return (Zero)" statement
430      * 3. A "Return empty buffer" statement
431      *
432      * We will return a buffer containing a single EndTag
433      * resource descriptor.
434      */
435     if (OriginalObject)
436     {
437         switch (OriginalObject->Common.Type)
438         {
439         case ACPI_TYPE_INTEGER:

441             /* We can only repair an Integer==0 */

443             if (OriginalObject->Integer.Value)
444             {
445                 return (AE_AML_OPERAND_TYPE);
446             }
447             break;

449         case ACPI_TYPE_BUFFER:

451             if (OriginalObject->Buffer.Length)
452             {
453                 /* Additional checks can be added in the future */

455                 *ReturnObject = NULL;
456                 return (AE_OK);
457             }
```



```
458             break;

460         case ACPI_TYPE_STRING:
461         default:

463             return (AE_AML_OPERAND_TYPE);
464         }
465     }

467     /* Create the new buffer object for the resource descriptor */

469     NewObject = AcpiUtCreateBufferObject (2);
470     if (!NewObject)
471     {
472         return (AE_NO_MEMORY);
473     }

475     Buffer = ACPI_CAST_PTR (UINT8, NewObject->Buffer.Pointer);

477     /* Initialize the Buffer with a single EndTag descriptor */

479     Buffer[0] = (ACPI_RESOURCE_NAME_END_TAG | ASL_RDESC_END_TAG_SIZE);
480     Buffer[1] = 0x00;

482     *ReturnObject = NewObject;
483     return (AE_OK);
484 }
```

```
**********************************************************
   26291 Thu Dec 26 13:49:17 2013
new/usr/src/common/acpica/components/namespace/nsdump.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsdump - table dumping routines for debug
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSDUMP_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acoutput.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsdump")

/* Local prototypes */

#ifdef ACPI_OBSOLETE_FUNCTIONS
void
```



```
AcpiNsDumpRootDevices (
    void);

static ACPI_STATUS
AcpiNsDumpOneDevice (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);
#endif


#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

static ACPI_STATUS
AcpiNsDumpOneObjectPath (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_STATUS
AcpiNsGetMaxDepth (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    AcpiNsPrintPathname
 *
 * PARAMETERS:  NumSegments         - Number of ACPI name segments
 *              Pathname            - The compressed (internal) path
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print an object's full namespace pathname
 *
 *****************************************************************************/

void
AcpiNsPrintPathname (
    UINT32                  NumSegments,
    char                    *Pathname)
{
    UINT32                  i;


    ACPI_FUNCTION_NAME (NsPrintPathname);


    /* Check if debug output enabled */

    if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_NAMES, ACPI_NAMESPACE))
    if (!(AcpiDbgLevel & ACPI_LV_NAMES) || !(AcpiDbgLayer & ACPI_NAMESPACE))
    {
        return;
    }

    /* Print the entire name */

    ACPI_DEBUG_PRINT ((ACPI_DB_NAMES, "["));

    while (NumSegments)
```

```
124     {
125         for (i = 0; i < 4; i++)
126         {
127             ACPI_IS_PRINT (Pathname[i]) ?
128                 AcpiOsPrintf ("%c", Pathname[i]) :
129                 AcpiOsPrintf ("?");
130         }

132         Pathname += ACPI_NAME_SIZE;
133         NumSegments--;
134         if (NumSegments)
135         {
136             AcpiOsPrintf (".");
137         }
138     }

140     AcpiOsPrintf ("]\n");
141 }


144 /*******************************************************************************
145  *
146  * FUNCTION:    AcpiNsDumpPathname
147  *
148  * PARAMETERS:  Handle              - Object
149  *              Msg                 - Prefix message
150  *              Level               - Desired debug level
151  *              Component           - Caller's component ID
152  *
153  * RETURN:      None
154  *
155  * DESCRIPTION: Print an object's full namespace pathname
156  *              Manages allocation/freeing of a pathname buffer
157  *
158  ******************************************************************************/

160 void
161 AcpiNsDumpPathname (
162     ACPI_HANDLE             Handle,
163     char                    *Msg,
164     UINT32                  Level,
165     UINT32                  Component)
166 {

168     ACPI_FUNCTION_TRACE (NsDumpPathname);


171     /* Do this only if the requested debug level and component are enabled */

173     if (!ACPI_IS_DEBUG_ENABLED (Level, Component))
154     if (!(AcpiDbgLevel & Level) || !(AcpiDbgLayer & Component))
174     {
175         return_VOID;
176     }

178     /* Convert handle to a full pathname and print it (with supplied message) */

180     AcpiNsPrintNodePathname (Handle, Msg);
181     AcpiOsPrintf ("\n");
182     return_VOID;
183 }


186 /*******************************************************************************
187  *
188  * FUNCTION:    AcpiNsDumpOneObject
```



```
189  *
190  * PARAMETERS:  ObjHandle           - Node to be dumped
191  *              Level               - Nesting level of the handle
192  *              Context             - Passed into WalkNamespace
193  *              ReturnValue         - Not used
194  *
195  * RETURN:      Status
196  *
197  * DESCRIPTION: Dump a single Node
198  *              This procedure is a UserFunction called by AcpiNsWalkNamespace.
199  *
200  ******************************************************************************/

202 ACPI_STATUS
203 AcpiNsDumpOneObject (
204     ACPI_HANDLE             ObjHandle,
205     UINT32                  Level,
206     void                    *Context,
207     void                    **ReturnValue)
208 {
209     ACPI_WALK_INFO          *Info = (ACPI_WALK_INFO *) Context;
210     ACPI_NAMESPACE_NODE     *ThisNode;
211     ACPI_OPERAND_OBJECT     *ObjDesc = NULL;
212     ACPI_OBJECT_TYPE        ObjType;
213     ACPI_OBJECT_TYPE        Type;
214     UINT32                  BytesToDump;
215     UINT32                  DbgLevel;
216     UINT32                  i;


219     ACPI_FUNCTION_NAME (NsDumpOneObject);


222     /* Is output enabled? */

224     if (!(AcpiDbgLevel & Info->DebugLevel))
225     {
226         return (AE_OK);
227     }

229     if (!ObjHandle)
230     {
231         ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Null object handle\n"));
232         return (AE_OK);
233     }

235     ThisNode = AcpiNsValidateHandle (ObjHandle);
236     if (!ThisNode)
237     {
238         ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Invalid object handle %p\n",
239             ObjHandle));
240         return (AE_OK);
241     }

243     Type = ThisNode->Type;

245     /* Check if the owner matches */

247     if ((Info->OwnerId != ACPI_OWNER_ID_MAX) &&
248         (Info->OwnerId != ThisNode->OwnerId))
249     {
250         return (AE_OK);
251     }

253     if (!(Info->DisplayType & ACPI_DISPLAY_SHORT))
254     {
```

```
255         /* Indent the object according to the level */
257         AcpiOsPrintf ("%2d%*s", (UINT32) Level - 1, (int) Level * 2, " ");
259         /* Check the node type and name */
261         if (Type > ACPI_TYPE_LOCAL_MAX)
262         {
263             ACPI_WARNING ((AE_INFO, "Invalid ACPI Object Type 0x%08X", Type));
264         }
266         AcpiOsPrintf ("%4.4s", AcpiUtGetNodeName (ThisNode));
267     }
269     /* Now we can print out the pertinent information */
271     AcpiOsPrintf (" %-12s %p %2.2X ",
272             AcpiUtGetTypeName (Type), ThisNode, ThisNode->OwnerId);
274     DbgLevel = AcpiDbgLevel;
275     AcpiDbgLevel = 0;
276     ObjDesc = AcpiNsGetAttachedObject (ThisNode);
277     AcpiDbgLevel = DbgLevel;
279     /* Temp nodes are those nodes created by a control method */
281     if (ThisNode->Flags & ANOBJ_TEMPORARY)
282     {
283         AcpiOsPrintf ("(T) ");
284     }
286     switch (Info->DisplayType & ACPI_DISPLAY_MASK)
287     {
288     case ACPI_DISPLAY_SUMMARY:
290         if (!ObjDesc)
291         {
292             /* No attached object. Some types should always have an object */
273             /* No attached object, we are done */
294             switch (Type)
295             {
296             case ACPI_TYPE_INTEGER:
297             case ACPI_TYPE_PACKAGE:
298             case ACPI_TYPE_BUFFER:
299             case ACPI_TYPE_STRING:
300             case ACPI_TYPE_METHOD:
302                 AcpiOsPrintf ("<No attached object>");
303                 break;
305             default:
307                 break;
308             }
310             AcpiOsPrintf ("\n");
311             return (AE_OK);
312         }
314         switch (Type)
315         {
316         case ACPI_TYPE_PROCESSOR:
318             AcpiOsPrintf ("ID %02X Len %02X Addr %p\n",
283             AcpiOsPrintf ("ID %X Len %.4X Addr %p\n",
```



```
319                 ObjDesc->Processor.ProcId, ObjDesc->Processor.Length,
320                 ACPI_CAST_PTR (void, ObjDesc->Processor.Address));
321             break;
323         case ACPI_TYPE_DEVICE:
325             AcpiOsPrintf ("Notify Object: %p\n", ObjDesc);
326             break;
328         case ACPI_TYPE_METHOD:
330             AcpiOsPrintf ("Args %X Len %.4X Aml %p\n",
331                 (UINT32) ObjDesc->Method.ParamCount,
332                 ObjDesc->Method.AmlLength, ObjDesc->Method.AmlStart);
333             break;
335         case ACPI_TYPE_INTEGER:
337             AcpiOsPrintf ("= %8.8X%8.8X\n",
338                 ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
339             break;
341         case ACPI_TYPE_PACKAGE:
343             if (ObjDesc->Common.Flags & AOPOBJ_DATA_VALID)
344             {
345                 AcpiOsPrintf ("Elements %.2X\n",
346                     ObjDesc->Package.Count);
347             }
348             else
349             {
350                 AcpiOsPrintf ("[Length not yet evaluated]\n");
351             }
352             break;
354         case ACPI_TYPE_BUFFER:
356             if (ObjDesc->Common.Flags & AOPOBJ_DATA_VALID)
357             {
358                 AcpiOsPrintf ("Len %.2X",
359                             ObjDesc->Buffer.Length);
361                 /* Dump some of the buffer */
363                 if (ObjDesc->Buffer.Length > 0)
364                 {
365                     AcpiOsPrintf (" =");
366                     for (i = 0; (i < ObjDesc->Buffer.Length && i < 12); i++)
367                     {
368                         AcpiOsPrintf (" %.2hX", ObjDesc->Buffer.Pointer[i]);
369                     }
370                 }
371                 AcpiOsPrintf ("\n");
372             }
373             else
374             {
375                 AcpiOsPrintf ("[Length not yet evaluated]\n");
376             }
377             break;
```

```
379         case ACPI_TYPE_STRING:

381             AcpiOsPrintf ("Len %.2X ", ObjDesc->String.Length);
382             AcpiUtPrintString (ObjDesc->String.Pointer, 32);
383             AcpiOsPrintf ("\n");
384             break;


386         case ACPI_TYPE_REGION:

388             AcpiOsPrintf ("[%s]",
389                 AcpiUtGetRegionName (ObjDesc->Region.SpaceId));
390             if (ObjDesc->Region.Flags & AOPOBJ_DATA_VALID)
391             {
392                 AcpiOsPrintf (" Addr %8.8X%8.8X Len %.4X\n",
393                     ACPI_FORMAT_NATIVE_UINT (ObjDesc->Region.Address),
394                     ObjDesc->Region.Length);
395             }
396             else
397             {
398                 AcpiOsPrintf (" [Address/Length not yet evaluated]\n");
399             }
400             break;


402         case ACPI_TYPE_LOCAL_REFERENCE:

404             AcpiOsPrintf ("[%s]\n", AcpiUtGetReferenceName (ObjDesc));
405             break;


407         case ACPI_TYPE_BUFFER_FIELD:

409             if (ObjDesc->BufferField.BufferObj &&
410                 ObjDesc->BufferField.BufferObj->Buffer.Node)
411             {
412                 AcpiOsPrintf ("Buf [%4.4s]",
413                     AcpiUtGetNodeName (
414                         ObjDesc->BufferField.BufferObj->Buffer.Node));
415             }
416             break;


418         case ACPI_TYPE_LOCAL_REGION_FIELD:

420             AcpiOsPrintf ("Rgn [%4.4s]",
421                 AcpiUtGetNodeName (
422                     ObjDesc->CommonField.RegionObj->Region.Node));
423             break;


425         case ACPI_TYPE_LOCAL_BANK_FIELD:

427             AcpiOsPrintf ("Rgn [%4.4s] Bnk [%4.4s]",
428                 AcpiUtGetNodeName (
429                     ObjDesc->CommonField.RegionObj->Region.Node),
430                 AcpiUtGetNodeName (
431                     ObjDesc->BankField.BankObj->CommonField.Node));
432             break;


434         case ACPI_TYPE_LOCAL_INDEX_FIELD:

436             AcpiOsPrintf ("Idx [%4.4s] Dat [%4.4s]",
437                 AcpiUtGetNodeName (
438                     ObjDesc->IndexField.IndexObj->CommonField.Node),
```



```
439                 AcpiUtGetNodeName (
440                     ObjDesc->IndexField.DataObj->CommonField.Node));
441             break;


443         case ACPI_TYPE_LOCAL_ALIAS:
444         case ACPI_TYPE_LOCAL_METHOD_ALIAS:

446             AcpiOsPrintf ("Target %4.4s (%p)\n",
447                 AcpiUtGetNodeName (ObjDesc), ObjDesc);
448             break;

450         default:

452             AcpiOsPrintf ("Object %p\n", ObjDesc);
453             break;
454         }

456         /* Common field handling */

458         switch (Type)
459         {
460         case ACPI_TYPE_BUFFER_FIELD:
461         case ACPI_TYPE_LOCAL_REGION_FIELD:
462         case ACPI_TYPE_LOCAL_BANK_FIELD:
463         case ACPI_TYPE_LOCAL_INDEX_FIELD:

465             AcpiOsPrintf (" Off %.3X Len %.2X Acc %.2hd\n",
466                 (ObjDesc->CommonField.BaseByteOffset * 8)
467                     + ObjDesc->CommonField.StartFieldBitOffset,
468                 ObjDesc->CommonField.BitLength,
469                 ObjDesc->CommonField.AccessByteWidth);
470             break;

472         default:

474             break;
475         }
476         break;


478     case ACPI_DISPLAY_OBJECTS:

480         AcpiOsPrintf ("O:%p", ObjDesc);
481         if (!ObjDesc)
482         {
483             /* No attached object, we are done */

485             AcpiOsPrintf ("\n");
486             return (AE_OK);
487         }

489         AcpiOsPrintf ("(R%u)", ObjDesc->Common.ReferenceCount);

491         switch (Type)
492         {
493         case ACPI_TYPE_METHOD:

495             /* Name is a Method and its AML offset/length are set */

497             AcpiOsPrintf (" M:%p-%X\n", ObjDesc->Method.AmlStart,
498                 ObjDesc->Method.AmlLength);
499             break;

501         case ACPI_TYPE_INTEGER:
```

```
503             AcpiOsPrintf (" I:%8.8X8.8%X\n",
504                 ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
505             break;

507         case ACPI_TYPE_STRING:

509             AcpiOsPrintf (" S:%p-%X\n", ObjDesc->String.Pointer,
510                 ObjDesc->String.Length);
511             break;

513         case ACPI_TYPE_BUFFER:

515             AcpiOsPrintf (" B:%p-%X\n", ObjDesc->Buffer.Pointer,
516                 ObjDesc->Buffer.Length);
517             break;

519         default:

521             AcpiOsPrintf ("\n");
522             break;
523         }
524         break;


526     default:
527         AcpiOsPrintf ("\n");
528         break;
529     }

531     /* If debug turned off, done */

533     if (!(AcpiDbgLevel & ACPI_LV_VALUES))
534     {
535         return (AE_OK);
536     }

538     /* If there is an attached object, display it */

540     DbgLevel     = AcpiDbgLevel;
541     AcpiDbgLevel = 0;
542     ObjDesc      = AcpiNsGetAttachedObject (ThisNode);
543     AcpiDbgLevel = DbgLevel;

545     /* Dump attached objects */

547     while (ObjDesc)
548     {
549         ObjType = ACPI_TYPE_INVALID;
550         AcpiOsPrintf ("Attached Object %p: ", ObjDesc);

552         /* Decode the type of attached object and dump the contents */

554         switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
555         {
556         case ACPI_DESC_TYPE_NAMED:

558             AcpiOsPrintf ("(Ptr to Node)\n");
559             BytesToDump = sizeof (ACPI_NAMESPACE_NODE);
560             ACPI_DUMP_BUFFER (ObjDesc, BytesToDump);
561             break;

563         case ACPI_DESC_TYPE_OPERAND:

565             ObjType = ObjDesc->Common.Type;

567             if (ObjType > ACPI_TYPE_LOCAL_MAX)
```



```
568             {
569                 AcpiOsPrintf ("(Pointer to ACPI Object type %.2X [UNKNOWN])\n",
570                     ObjType);
571                 BytesToDump = 32;
572             }
573             else
574             {
575                 AcpiOsPrintf ("(Pointer to ACPI Object type %.2X [%s])\n",
576                     ObjType, AcpiUtGetTypeName (ObjType));
577                 BytesToDump = sizeof (ACPI_OPERAND_OBJECT);
578             }

580             ACPI_DUMP_BUFFER (ObjDesc, BytesToDump);
581             break;

583         default:

585             break;
586         }

588         /* If value is NOT an internal object, we are done */

590         if (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND)
591         {
592             goto Cleanup;
593         }

595         /* Valid object, get the pointer to next level, if any */

597         switch (ObjType)
598         {
599         case ACPI_TYPE_BUFFER:
600         case ACPI_TYPE_STRING:
601             /*
602              * NOTE: takes advantage of common fields between string/buffer
603              */
604             BytesToDump = ObjDesc->String.Length;
605             ObjDesc = (void *) ObjDesc->String.Pointer;
606             AcpiOsPrintf ( "(Buffer/String pointer %p length %X)\n",
607                 ObjDesc, BytesToDump);
608             ACPI_DUMP_BUFFER (ObjDesc, BytesToDump);
609             goto Cleanup;

611         case ACPI_TYPE_BUFFER_FIELD:

613             ObjDesc = (ACPI_OPERAND_OBJECT *) ObjDesc->BufferField.BufferObj;
614             break;

616         case ACPI_TYPE_PACKAGE:

618             ObjDesc = (void *) ObjDesc->Package.Elements;
619             break;

621         case ACPI_TYPE_METHOD:

623             ObjDesc = (void *) ObjDesc->Method.AmlStart;
624             break;

626         case ACPI_TYPE_LOCAL_REGION_FIELD:

628             ObjDesc = (void *) ObjDesc->Field.RegionObj;
629             break;

631         case ACPI_TYPE_LOCAL_BANK_FIELD:

633             ObjDesc = (void *) ObjDesc->BankField.RegionObj;
```

```
634             break;

636         case ACPI_TYPE_LOCAL_INDEX_FIELD:

638             ObjDesc = (void *) ObjDesc->IndexField.IndexObj;
639             break;

641         default:

643             goto Cleanup;
644         }

646         ObjType = ACPI_TYPE_INVALID;   /* Terminate loop after next pass */
647     }

649 Cleanup:
650     AcpiOsPrintf ("\n");
651     return (AE_OK);
652 }
```

*unchanged_portion_omitted_*

```
714 /******************************************************************************
715  *
716  * FUNCTION:    AcpiNsDumpOneObjectPath, AcpiNsGetMaxDepth
717  *
718  * PARAMETERS:  ObjHandle           - Node to be dumped
719  *              Level               - Nesting level of the handle
720  *              Context             - Passed into WalkNamespace
721  *              ReturnValue         - Not used
722  *
723  * RETURN:      Status
724  *
725  * DESCRIPTION: Dump the full pathname to a namespace object. AcpNsGetMaxDepth
726  *              computes the maximum nesting depth in the namespace tree, in
727  *              order to simplify formatting in AcpiNsDumpOneObjectPath.
728  *              These procedures are UserFunctions called by AcpiNsWalkNamespace
729  *
730  *****************************************************************************/

732 static ACPI_STATUS
733 AcpiNsDumpOneObjectPath (
734     ACPI_HANDLE             ObjHandle,
735     UINT32                  Level,
736     void                    *Context,
737     void                    **ReturnValue)
738 {
739     UINT32                  MaxLevel = *((UINT32 *) Context);
740     char                    *Pathname;
741     ACPI_NAMESPACE_NODE     *Node;
742     int                     PathIndent;


745     if (!ObjHandle)
746     {
747         return (AE_OK);
748     }

750     Node = AcpiNsValidateHandle (ObjHandle);
751     if (!Node)
752     {
753         /* Ignore bad node during namespace walk */

755         return (AE_OK);
756     }
```



```
758     Pathname = AcpiNsGetExternalPathname (Node);

760     PathIndent = 1;
761     if (Level <= MaxLevel)
762     {
763         PathIndent = MaxLevel - Level + 1;
764     }

766     AcpiOsPrintf ("%2d%*s%-12s%*s",
767         Level, Level, " ", AcpiUtGetTypeName (Node->Type),
768         PathIndent, " ");

770     AcpiOsPrintf ("%s\n", &Pathname[1]);
771     ACPI_FREE (Pathname);
772     return (AE_OK);
773 }


776 static ACPI_STATUS
777 AcpiNsGetMaxDepth (
778     ACPI_HANDLE             ObjHandle,
779     UINT32                  Level,
780     void                    *Context,
781     void                    **ReturnValue)
782 {
783     UINT32                  *MaxLevel = (UINT32 *) Context;


786     if (Level > *MaxLevel)
787     {
788         *MaxLevel = Level;
789     }
790     return (AE_OK);
791 }


794 /******************************************************************************
795  *
796  * FUNCTION:    AcpiNsDumpObjectPaths
797  *
798  * PARAMETERS:  Type                - Object type to be dumped
799  *              DisplayType         - 0 or ACPI_DISPLAY_SUMMARY
800  *              MaxDepth            - Maximum depth of dump. Use ACPI_UINT32_MAX
801  *                                    for an effectively unlimited depth.
802  *              OwnerId             - Dump only objects owned by this ID. Use
803  *                                    ACPI_UINT32_MAX to match all owners.
804  *              StartHandle         - Where in namespace to start/end search
805  *
806  * RETURN:      None
807  *
808  * DESCRIPTION: Dump full object pathnames within the loaded namespace. Uses
809  *              AcpiNsWalkNamespace in conjunction with AcpiNsDumpOneObjectPath.
810  *
811  *****************************************************************************/

813 void
814 AcpiNsDumpObjectPaths (
815     ACPI_OBJECT_TYPE        Type,
816     UINT8                   DisplayType,
817     UINT32                  MaxDepth,
818     ACPI_OWNER_ID           OwnerId,
819     ACPI_HANDLE             StartHandle)
820 {
821     ACPI_STATUS             Status;
822     UINT32                  MaxLevel = 0;
```

```
825     ACPI_FUNCTION_ENTRY ();


828     /*
829      * Just lock the entire namespace for the duration of the dump.
830      * We don't want any changes to the namespace during this time,
831      * especially the temporary nodes since we are going to display
832      * them also.
833      */
834     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
835     if (ACPI_FAILURE (Status))
836     {
837         AcpiOsPrintf ("Could not acquire namespace mutex\n");
838         return;
839     }

841     /* Get the max depth of the namespace tree, for formatting later */

843     (void) AcpiNsWalkNamespace (Type, StartHandle, MaxDepth,
844                 ACPI_NS_WALK_NO_UNLOCK | ACPI_NS_WALK_TEMP_NODES,
845                 AcpiNsGetMaxDepth, NULL, (void *) &MaxLevel, NULL);

847     /* Now dump the entire namespace */

849     (void) AcpiNsWalkNamespace (Type, StartHandle, MaxDepth,
850                 ACPI_NS_WALK_NO_UNLOCK | ACPI_NS_WALK_TEMP_NODES,
851                 AcpiNsDumpOneObjectPath, NULL, (void *) &MaxLevel, NULL);

853     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
854 }


857 /*******************************************************************************
858  *
859  * FUNCTION:    AcpiNsDumpEntry
860  *
861  * PARAMETERS:  Handle              - Node to be dumped
862  *              DebugLevel          - Output level
863  *
864  * RETURN:      None
865  *
866  * DESCRIPTION: Dump a single Node
867  *
868  ******************************************************************************/

870 void
871 AcpiNsDumpEntry (
872     ACPI_HANDLE             Handle,
873     UINT32                  DebugLevel)
874 {
875     ACPI_WALK_INFO          Info;


878     ACPI_FUNCTION_ENTRY ();


881     Info.DebugLevel = DebugLevel;
882     Info.OwnerId = ACPI_OWNER_ID_MAX;
883     Info.DisplayType = ACPI_DISPLAY_SUMMARY;

885     (void) AcpiNsDumpOneObject (Handle, 1, &Info, NULL);
886 }
```

*unchanged_portion_omitted_*

```
938 #endif
939 #endif
```

****
5063 Thu Dec 26 13:49:17 2013
new/usr/src/common/acpica/components/namespace/nsdumpdv.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: nsdump - table dumping routines for debug
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __NSDUMPDV_C__

  46 #include "acpi.h"


  49 /* TBD: This entire module is apparently obsolete and should be removed */

  51 #define _COMPONENT          ACPI_NAMESPACE
  52         ACPI_MODULE_NAME    ("nsdumpdv")

  54 #ifdef ACPI_OBSOLETE_FUNCTIONS
  55 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

  57 #include "acnamesp.h"

  59 /******************************************************************************
```



```
  60  *
  61  * FUNCTION:    AcpiNsDumpOneDevice
  62  *
  63  * PARAMETERS:  Handle              - Node to be dumped
  64  *              Level               - Nesting level of the handle
  65  *              Context             - Passed into WalkNamespace
  66  *              ReturnValue         - Not used
  67  *
  68  * RETURN:      Status
  69  *
  70  * DESCRIPTION: Dump a single Node that represents a device
  71  *              This procedure is a UserFunction called by AcpiNsWalkNamespace.
  72  *
  73  *****************************************************************************/

  75 static ACPI_STATUS
  76 AcpiNsDumpOneDevice (
  77     ACPI_HANDLE             ObjHandle,
  78     UINT32                  Level,
  79     void                    *Context,
  80     void                    **ReturnValue)
  81 {
  82     ACPI_BUFFER             Buffer;
  83     ACPI_DEVICE_INFO        *Info;
  84     ACPI_STATUS             Status;
  85     UINT32                  i;


  88     ACPI_FUNCTION_NAME (NsDumpOneDevice);


  91     Status = AcpiNsDumpOneObject (ObjHandle, Level, Context, ReturnValue);

  93     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
  94     Status = AcpiGetObjectInfo (ObjHandle, &Buffer);
  95     if (ACPI_SUCCESS (Status))
  96     {
  97         Info = Buffer.Pointer;
  98         for (i = 0; i < Level; i++)
  99         {
 100             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_TABLES, " "));
 101         }

 103         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_TABLES,
 104             "    HID: %s, ADR: %8.8X%8.8X, Status: %X\n",
 105             Info->HardwareId.Value, ACPI_FORMAT_UINT64 (Info->Address),
 106             Info->CurrentStatus));
 107         ACPI_FREE (Info);
 108     }

 110     return (Status);
 111 }


 114 /******************************************************************************
 115  *
 116  * FUNCTION:    AcpiNsDumpRootDevices
 117  *
 118  * PARAMETERS:  None
 119  *
 120  * RETURN:      None
 121  *
 122  * DESCRIPTION: Dump all objects of type "device"
 123  *
 124  *****************************************************************************/
```

```
126 void
127 AcpiNsDumpRootDevices (
128     void)
129 {
130     ACPI_HANDLE             SysBusHandle;
131     ACPI_STATUS             Status;


134     ACPI_FUNCTION_NAME (NsDumpRootDevices);


137     /* Only dump the table if tracing is enabled */

139     if (!(ACPI_LV_TABLES & AcpiDbgLevel))
140     {
141         return;
142     }

144     Status = AcpiGetHandle (NULL, METHOD_NAME__SB_, &SysBusHandle);
144     Status = AcpiGetHandle (NULL, ACPI_NS_SYSTEM_BUS, &SysBusHandle);
145     if (ACPI_FAILURE (Status))
146     {
147         return;
148     }

150     ACPI_DEBUG_PRINT ((ACPI_DB_TABLES,
151         "Display of all devices in the namespace:\n"));

153     Status = AcpiNsWalkNamespace (ACPI_TYPE_DEVICE, SysBusHandle,
154                 ACPI_UINT32_MAX, ACPI_NS_WALK_NO_UNLOCK,
155                 AcpiNsDumpOneDevice, NULL, NULL, NULL);
156 }

158 #endif
159 #endif
```

```
**********************************************************
   16392 Thu Dec 26 13:49:18 2013
new/usr/src/common/acpica/components/namespace/nseval.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: nseval - Object evaluation, includes control method execution
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSEVAL_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acinterp.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nseval")

/* Local prototypes */

static void
AcpiNsExecModuleCode (
    ACPI_OPERAND_OBJECT     *MethodObj,
```



```
    ACPI_EVALUATE_INFO      *Info);


/******************************************************************************
 *
 * FUNCTION:    AcpiNsEvaluate
 *
 * PARAMETERS:  Info            - Evaluation info block, contains:
 *                  PrefixNode      - Prefix or Method/Object Node to execute
 *                  RelativePath    - Name of method to execute, If NULL, the
 *                                    Node is the object to execute
 *                  Parameters      - List of parameters to pass to the method,
 *                                    terminated by NULL. Params itself may be
 *                                    NULL if no parameters are being passed.
 *                  ReturnObject    - Where to put method's return value (if
 *                                    any). If NULL, no value is returned.
 *                  ParameterType   - Type of Parameter list
 *                  ReturnObject    - Where to put method's return value (if
 *                                    any). If NULL, no value is returned.
 *                  Flags           - ACPI_IGNORE_RETURN_VALUE to delete return
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute a control method or return the current value of an
 *              ACPI namespace object.
 *
 * MUTEX:       Locks interpreter
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsEvaluate (
    ACPI_EVALUATE_INFO      *Info)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (NsEvaluate);


    if (!Info)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (!Info->Node)
    {
        /*
         * Get the actual namespace node for the target object if we
         * need to. Handles these cases:
         *
         * 1) Null node, valid pathname from root (absolute path)
         * 2) Node and valid pathname (path relative to Node)
         * 3) Node, Null pathname
         */
        Status = AcpiNsGetNode (Info->PrefixNode, Info->RelativePathname,
            ACPI_NS_NO_UPSEARCH, &Info->Node);
        if (ACPI_FAILURE (Status))
        {
            return_ACPI_STATUS (Status);
        }
    }

    /*
     * For a method alias, we must grab the actual method node so that
     * proper scoping context will be established before execution.
```

```
127      */
128     if (AcpiNsGetType (Info->Node) == ACPI_TYPE_LOCAL_METHOD_ALIAS)
129     {
130         Info->Node = ACPI_CAST_PTR (
131             ACPI_NAMESPACE_NODE, Info->Node->Object);
132     }

134     /* Complete the info block initialization */

136     Info->ReturnObject = NULL;
137     Info->NodeFlags = Info->Node->Flags;
138     Info->ObjDesc = AcpiNsGetAttachedObject (Info->Node);

140     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES, "%s [%p] Value %p\n",
141         Info->RelativePathname, Info->Node,
142         AcpiNsGetAttachedObject (Info->Node)));

144     /* Get info if we have a predefined name (_HID, etc.) */

146     Info->Predefined = AcpiUtMatchPredefinedMethod (Info->Node->Name.Ascii);

148     /* Get the full pathname to the object, for use in warning messages */

150     Info->FullPathname = AcpiNsGetExternalPathname (Info->Node);
151     if (!Info->FullPathname)
152     {
153         return_ACPI_STATUS (AE_NO_MEMORY);
154     }

156     /* Count the number of arguments being passed in */

158     Info->ParamCount = 0;
159     if (Info->Parameters)
160     {
161         while (Info->Parameters[Info->ParamCount])
162         {
163             Info->ParamCount++;
164         }

166         /* Warn on impossible argument count */

168         if (Info->ParamCount > ACPI_METHOD_NUM_ARGS)
169         {
170             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, ACPI_WARN_ALWAYS
171                 "Excess arguments (%u) - using only %u",
172                 Info->ParamCount, ACPI_METHOD_NUM_ARGS));

174             Info->ParamCount = ACPI_METHOD_NUM_ARGS;
175         }
176     }

178     /*
179      * For predefined names: Check that the declared argument count
180      * matches the ACPI spec -- otherwise this is a BIOS error.
181      */
182     AcpiNsCheckAcpiCompliance (Info->FullPathname, Info->Node,
183         Info->Predefined);

185     /*
186      * For all names: Check that the incoming argument count for
187      * this method/object matches the actual ASL/AML definition.
188      */
189     AcpiNsCheckArgumentCount (Info->FullPathname, Info->Node,
190         Info->ParamCount, Info->Predefined);

192     /* For predefined names: Typecheck all incoming arguments */
```



```
194     AcpiNsCheckArgumentTypes (Info);

196     /*
197      * Three major evaluation cases:
198      *
199      * 1) Object types that cannot be evaluated by definition
200      * 2) The object is a control method -- execute it
201      * 3) The object is not a method -- just return it's current value
202      */
203     switch (AcpiNsGetType (Info->Node))
204     {
205     case ACPI_TYPE_DEVICE:
206     case ACPI_TYPE_EVENT:
207     case ACPI_TYPE_MUTEX:
208     case ACPI_TYPE_REGION:
209     case ACPI_TYPE_THERMAL:
210     case ACPI_TYPE_LOCAL_SCOPE:
211         /*
212          * 1) Disallow evaluation of certain object types. For these,
213          *    object evaluation is undefined and not supported.
214          */
215         ACPI_ERROR ((AE_INFO,
216             "%s: Evaluation of object type [%s] is not supported",
217             Info->FullPathname,
218             AcpiUtGetTypeName (Info->Node->Type)));

220         Status = AE_TYPE;
221         goto Cleanup;

223     case ACPI_TYPE_METHOD:
224         /*
225          * 2) Object is a control method - execute it
226          */

228         /* Verify that there is a method object associated with this node */

230         if (!Info->ObjDesc)
231         {
232             ACPI_ERROR ((AE_INFO, "%s: Method has no attached sub-object",
233                 Info->FullPathname));
234             Status = AE_NULL_OBJECT;
235             goto Cleanup;
236         }

238         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
239             "**** Execute method [%s] at AML address %p length %X\n",
240             Info->FullPathname,
241             Info->ObjDesc->Method.AmlStart + 1,
242             Info->ObjDesc->Method.AmlLength - 1));

244         /*
245          * Any namespace deletion must acquire both the namespace and
246          * interpreter locks to ensure that no thread is using the portion of
247          * the namespace that is being deleted.
248          *
249          * Execute the method via the interpreter. The interpreter is locked
250          * here before calling into the AML parser
251          */
252         AcpiExEnterInterpreter ();
253         Status = AcpiPsExecuteMethod (Info);
254         AcpiExExitInterpreter ();
255         break;

257     default:
258         /*
```

```
259          * 3) All other non-method objects -- get the current object value
260          */
262         /*
263          * Some objects require additional resolution steps (e.g., the Node
264          * may be a field that must be read, etc.) -- we can't just grab
265          * the object out of the node.
266          *
267          * Use ResolveNodeToValue() to get the associated value.
268          *
269          * NOTE: we can get away with passing in NULL for a walk state because
270          * the Node is guaranteed to not be a reference to either a method
271          * local or a method argument (because this interface is never called
272          * from a running method.)
273          *
274          * Even though we do not directly invoke the interpreter for object
275          * resolution, we must lock it because we could access an OpRegion.
276          * The OpRegion access code assumes that the interpreter is locked.
277          */
278         AcpiExEnterInterpreter ();
280         /* TBD: ResolveNodeToValue has a strange interface, fix */
282         Info->ReturnObject = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Info->Node);
284         Status = AcpiExResolveNodeToValue (ACPI_CAST_INDIRECT_PTR (
285             ACPI_NAMESPACE_NODE, &Info->ReturnObject), NULL);
286         AcpiExExitInterpreter ();
288         if (ACPI_FAILURE (Status))
289         {
290             goto Cleanup;
291         }
293         ACPI_DEBUG_PRINT ((ACPI_DB_NAMES, "Returned object %p [%s]\n",
294             Info->ReturnObject,
295             AcpiUtGetObjectTypeName (Info->ReturnObject)));
297         Status = AE_CTRL_RETURN_VALUE; /* Always has a "return value" */
298         break;
299     }
301     /*
302      * For predefined names, check the return value against the ACPI
303      * specification. Some incorrect return value types are repaired.
304      */
305     (void) AcpiNsCheckReturnValue (Info->Node, Info, Info->ParamCount,
306         Status, &Info->ReturnObject);
308     /* Check if there is a return value that must be dealt with */
310     if (Status == AE_CTRL_RETURN_VALUE)
311     {
312         /* If caller does not want the return value, delete it */
314         if (Info->Flags & ACPI_IGNORE_RETURN_VALUE)
315         {
316             AcpiUtRemoveReference (Info->ReturnObject);
317             Info->ReturnObject = NULL;
318         }
320         /* Map AE_CTRL_RETURN_VALUE to AE_OK, we are done with it */
322         Status = AE_OK;
323     }
```



```
325     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
326         "*** Completed evaluation of object %s ***\n",
327         Info->RelativePathname));
329 Cleanup:
330     /*
331      * Namespace was unlocked by the handling AcpiNs* function, so we
332      * just free the pathname and return
333      */
334     ACPI_FREE (Info->FullPathname);
335     Info->FullPathname = NULL;
336     return_ACPI_STATUS (Status);
337 }
340 /*******************************************************************************
341  *
342  * FUNCTION:    AcpiNsExecModuleCodeList
343  *
344  * PARAMETERS:  None
345  *
346  * RETURN:      None. Exceptions during method execution are ignored, since
347  *              we cannot abort a table load.
348  *
349  * DESCRIPTION: Execute all elements of the global module-level code list.
350  *              Each element is executed as a single control method.
351  *
352  ******************************************************************************/
354 void
355 AcpiNsExecModuleCodeList (
356     void)
357 {
358     ACPI_OPERAND_OBJECT     *Prev;
359     ACPI_OPERAND_OBJECT     *Next;
360     ACPI_EVALUATE_INFO      *Info;
361     UINT32                  MethodCount = 0;
364     ACPI_FUNCTION_TRACE (NsExecModuleCodeList);
367     /* Exit now if the list is empty */
369     Next = AcpiGbl_ModuleCodeList;
370     if (!Next)
371     {
372         return_VOID;
373     }
375     /* Allocate the evaluation information block */
377     Info = ACPI_ALLOCATE (sizeof (ACPI_EVALUATE_INFO));
378     if (!Info)
379     {
380         return_VOID;
381     }
383     /* Walk the list, executing each "method" */
385     while (Next)
386     {
387         Prev = Next;
388         Next = Next->Method.Mutex;
390         /* Clear the link field and execute the method */
```

```
392         Prev->Method.Mutex = NULL;
393         AcpiNsExecModuleCode (Prev, Info);
394         MethodCount++;

396         /* Delete the (temporary) method object */

398         AcpiUtRemoveReference (Prev);
399     }

401     ACPI_INFO ((AE_INFO,
402         "Executed %u blocks of module-level executable AML code",
403         MethodCount));

405     ACPI_FREE (Info);
406     AcpiGbl_ModuleCodeList = NULL;
407     return_VOID;
408 }


411 /*******************************************************************************
412  *
413  * FUNCTION:    AcpiNsExecModuleCode
414  *
415  * PARAMETERS:  MethodObj           - Object container for the module-level code
416  *              Info                - Info block for method evaluation
417  *
418  * RETURN:      None. Exceptions during method execution are ignored, since
419  *              we cannot abort a table load.
420  *
421  * DESCRIPTION: Execute a control method containing a block of module-level
422  *              executable AML code. The control method is temporarily
423  *              installed to the root node, then evaluated.
424  *
425  ******************************************************************************/

427 static void
428 AcpiNsExecModuleCode (
429     ACPI_OPERAND_OBJECT     *MethodObj,
430     ACPI_EVALUATE_INFO      *Info)
431 {
432     ACPI_OPERAND_OBJECT     *ParentObj;
433     ACPI_NAMESPACE_NODE     *ParentNode;
434     ACPI_OBJECT_TYPE        Type;
435     ACPI_STATUS             Status;


438     ACPI_FUNCTION_TRACE (NsExecModuleCode);


441     /*
442      * Get the parent node. We cheat by using the NextObject field
443      * of the method object descriptor.
444      */
445     ParentNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
446                     MethodObj->Method.NextObject);
447     Type = AcpiNsGetType (ParentNode);

449     /*
450      * Get the region handler and save it in the method object. We may need
451      * this if an operation region declaration causes a _REG method to be run.
452      *
453      * We can't do this in AcpiPsLinkModuleCode because
454      * AcpiGbl_RootNode->Object is NULL at PASS1.
455      */
456     if ((Type == ACPI_TYPE_DEVICE) && ParentNode->Object)
```



```
457     {
458         MethodObj->Method.Dispatch.Handler =
459             ParentNode->Object->Device.Handler;
460     }

462     /* Must clear NextObject (AcpiNsAttachObject needs the field) */

464     MethodObj->Method.NextObject = NULL;

466     /* Initialize the evaluation information block */

468     ACPI_MEMSET (Info, 0, sizeof (ACPI_EVALUATE_INFO));
469     Info->PrefixNode = ParentNode;

471     /*
472      * Get the currently attached parent object. Add a reference, because the
473      * ref count will be decreased when the method object is installed to
474      * the parent node.
475      */
476     ParentObj = AcpiNsGetAttachedObject (ParentNode);
477     if (ParentObj)
478     {
479         AcpiUtAddReference (ParentObj);
480     }

482     /* Install the method (module-level code) in the parent node */

484     Status = AcpiNsAttachObject (ParentNode, MethodObj,
485                 ACPI_TYPE_METHOD);
486     if (ACPI_FAILURE (Status))
487     {
488         goto Exit;
489     }

491     /* Execute the parent node as a control method */

493     Status = AcpiNsEvaluate (Info);

495     ACPI_DEBUG_PRINT ((ACPI_DB_INIT, "Executed module-level code at %p\n",
496         MethodObj->Method.AmlStart));

498     /* Delete a possible implicit return value (in slack mode) */

500     if (Info->ReturnObject)
501     {
502         AcpiUtRemoveReference (Info->ReturnObject);
503     }

505     /* Detach the temporary method object */

507     AcpiNsDetachObject (ParentNode);

509     /* Restore the original parent object */

511     if (ParentObj)
512     {
513         Status = AcpiNsAttachObject (ParentNode, ParentObj, Type);
514     }
515     else
516     {
517         ParentNode->Type = (UINT8) Type;
518     }

520 Exit:
521     if (ParentObj)
522     {
```

```
523         AcpiUtRemoveReference (ParentObj);
524     }
525     return_VOID;
526 }
```

****
19552 Thu Dec 26 13:49:18 2013
new/usr/src/common/acpica/components/namespace/nsinit.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: nsinit - namespace initialization
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSXFINIT_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acdispat.h"
#include "acinterp.h"

#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsinit")

/* Local prototypes */

static ACPI_STATUS
AcpiNsInitOneObject (
```



```
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_STATUS
AcpiNsInitOneDevice (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_STATUS
AcpiNsFindIniMethods (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);


/******************************************************************************
 *
 * FUNCTION:    AcpiNsInitializeObjects
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk the entire namespace and perform any necessary
 *              initialization on the objects found therein
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsInitializeObjects (
    void)
{
    ACPI_STATUS             Status;
    ACPI_INIT_WALK_INFO     Info;


    ACPI_FUNCTION_TRACE (NsInitializeObjects);


    ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
        "***** Starting initialization of namespace objects ****\n"));
    ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
        "Completing Region/Field/Buffer/Package initialization:\n"));
        "Completing Region/Field/Buffer/Package initialization:"));

    /* Set all init info to zero */

    ACPI_MEMSET (&Info, 0, sizeof (ACPI_INIT_WALK_INFO));

    /* Walk entire namespace from the supplied root */

    Status = AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
                ACPI_UINT32_MAX, AcpiNsInitOneObject, NULL,
                &Info, NULL);
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "During WalkNamespace"));
    }

    ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
        "    Initialized %u/%u Regions %u/%u Fields %u/%u "
```

```
124         "\nInitialized %u/%u Regions %u/%u Fields %u/%u "
125         "Buffers %u/%u Packages (%u nodes)\n",
126         Info.OpRegionInit,  Info.OpRegionCount,
127         Info.FieldInit,     Info.FieldCount,
128         Info.BufferInit,    Info.BufferCount,
129         Info.PackageInit,   Info.PackageCount, Info.ObjectCount));

131     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
132         "%u Control Methods found\n", Info.MethodCount));
133     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
134         "%u Op Regions found\n", Info.OpRegionCount));

136     return_ACPI_STATUS (AE_OK);
137 }


140 /******************************************************************************
141  *
142  * FUNCTION:    AcpiNsInitializeDevices
143  *
144  * PARAMETERS:  None
145  *
146  * RETURN:      ACPI_STATUS
147  *
148  * DESCRIPTION: Walk the entire namespace and initialize all ACPI devices.
149  *              This means running _INI on all present devices.
150  *
151  *              Note: We install PCI config space handler on region access,
152  *              not here.
153  *
154  *****************************************************************************/

156 ACPI_STATUS
157 AcpiNsInitializeDevices (
158     void)
159 {
160     ACPI_STATUS             Status;
161     ACPI_DEVICE_WALK_INFO   Info;


164     ACPI_FUNCTION_TRACE (NsInitializeDevices);


167     /* Init counters */

169     Info.DeviceCount = 0;
170     Info.Num_STA = 0;
171     Info.Num_INI = 0;

173     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
174         "Initializing Device/Processor/Thermal objects "
175         "and executing _INI/_STA methods:\n"));
175         "by executing _INI methods:"));

177     /* Tree analysis: find all subtrees that contain _INI methods */

179     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
180                 ACPI_UINT32_MAX, FALSE, AcpiNsFindIniMethods, NULL, &Info, NULL)
181     if (ACPI_FAILURE (Status))
182     {
183         goto ErrorExit;
184     }

186     /* Allocate the evaluation information block */

188     Info.EvaluateInfo = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
```



```
189     if (!Info.EvaluateInfo)
190     {
191         Status = AE_NO_MEMORY;
192         goto ErrorExit;
193     }

195     /*
196      * Execute the "global" _INI method that may appear at the root. This
197      * support is provided for Windows compatibility (Vista+) and is not
198      * part of the ACPI specification.
199      */
200     Info.EvaluateInfo->PrefixNode = AcpiGbl_RootNode;
201     Info.EvaluateInfo->RelativePathname = METHOD_NAME__INI;
201     Info.EvaluateInfo->Pathname = METHOD_NAME__INI;
202     Info.EvaluateInfo->Parameters = NULL;
203     Info.EvaluateInfo->Flags = ACPI_IGNORE_RETURN_VALUE;

205     Status = AcpiNsEvaluate (Info.EvaluateInfo);
206     if (ACPI_SUCCESS (Status))
207     {
208         Info.Num_INI++;
209     }

211     /* Walk namespace to execute all _INIs on present devices */

213     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
214                 ACPI_UINT32_MAX, FALSE, AcpiNsInitOneDevice, NULL, &Info, NULL);

216     /*
217      * Any _OSI requests should be completed by now. If the BIOS has
218      * requested any Windows OSI strings, we will always truncate
219      * I/O addresses to 16 bits -- for Windows compatibility.
220      */
221     if (AcpiGbl_OsiData >= ACPI_OSI_WIN_2000)
222     {
223         AcpiGbl_TruncateIoAddresses = TRUE;
224     }

226     ACPI_FREE (Info.EvaluateInfo);
227     if (ACPI_FAILURE (Status))
228     {
229         goto ErrorExit;
230     }

232     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
233         "    Executed %u _INI methods requiring %u _STA executions "
233         "\nExecuted %u _INI methods requiring %u _STA executions "
234         "(examined %u objects)\n",
235         Info.Num_INI, Info.Num_STA, Info.DeviceCount));

237     return_ACPI_STATUS (Status);


240 ErrorExit:
241     ACPI_EXCEPTION ((AE_INFO, Status, "During device initialization"));
242     return_ACPI_STATUS (Status);
243 }


246 /******************************************************************************
247  *
248  * FUNCTION:    AcpiNsInitOneObject
249  *
250  * PARAMETERS:  ObjHandle       - Node
251  *              Level           - Current nesting level
252  *              Context         - Points to a init info struct
```

```
 *              ReturnValue     - Not used
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Callback from AcpiWalkNamespace. Invoked for every object
 *              within the  namespace.
 *
 *              Currently, the only objects that require initialization are:
 *              1) Methods
 *              2) Op Regions
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiNsInitOneObject (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue)
{
    ACPI_OBJECT_TYPE        Type;
    ACPI_STATUS             Status = AE_OK;
    ACPI_INIT_WALK_INFO     *Info = (ACPI_INIT_WALK_INFO *) Context;
    ACPI_NAMESPACE_NODE     *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
    ACPI_OPERAND_OBJECT     *ObjDesc;


    ACPI_FUNCTION_NAME (NsInitOneObject);


    Info->ObjectCount++;

    /* And even then, we are only interested in a few object types */

    Type = AcpiNsGetType (ObjHandle);
    ObjDesc = AcpiNsGetAttachedObject (Node);
    if (!ObjDesc)
    {
        return (AE_OK);
    }

    /* Increment counters for object types we are looking for */

    switch (Type)
    {
    case ACPI_TYPE_REGION:

        Info->OpRegionCount++;
        break;

    case ACPI_TYPE_BUFFER_FIELD:

        Info->FieldCount++;
        break;

    case ACPI_TYPE_LOCAL_BANK_FIELD:

        Info->FieldCount++;
        break;

    case ACPI_TYPE_BUFFER:

        Info->BufferCount++;
        break;

    case ACPI_TYPE_PACKAGE:
```



```
        Info->PackageCount++;
        break;

    default:

        /* No init required, just exit now */

        return (AE_OK);
    }

    /* If the object is already initialized, nothing else to do */

    if (ObjDesc->Common.Flags & AOPOBJ_DATA_VALID)
    {
        return (AE_OK);
    }

    /* Must lock the interpreter before executing AML code */

    AcpiExEnterInterpreter ();

    /*
     * Each of these types can contain executable AML code within the
     * declaration.
     */
    switch (Type)
    {
    case ACPI_TYPE_REGION:

        Info->OpRegionInit++;
        Status = AcpiDsGetRegionArguments (ObjDesc);
        break;

    case ACPI_TYPE_BUFFER_FIELD:

        Info->FieldInit++;
        Status = AcpiDsGetBufferFieldArguments (ObjDesc);
        break;

    case ACPI_TYPE_LOCAL_BANK_FIELD:

        Info->FieldInit++;
        Status = AcpiDsGetBankFieldArguments (ObjDesc);
        break;

    case ACPI_TYPE_BUFFER:

        Info->BufferInit++;
        Status = AcpiDsGetBufferArguments (ObjDesc);
        break;

    case ACPI_TYPE_PACKAGE:

        Info->PackageInit++;
        Status = AcpiDsGetPackageArguments (ObjDesc);
        break;

    default:

        /* No other types can get here */

        break;
    }

    if (ACPI_FAILURE (Status))
```

```
385     {
386         ACPI_EXCEPTION ((AE_INFO, Status,
387             "Could not execute arguments for [%4.4s] (%s)",
388             AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Type)));
389     }

391     /*
384      * Print a dot for each object unless we are going to print the entire
385      * pathname
386      */
387     if (!(AcpiDbgLevel & ACPI_LV_INIT_NAMES))
388     {
389         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT, "."));
390     }

392     /*
392      * We ignore errors from above, and always return OK, since we don't want
393      * to abort the walk on any single error.
394      */
395     AcpiExExitInterpreter ();
396     return (AE_OK);
397 }


400 /******************************************************************************
401  *
402  * FUNCTION:    AcpiNsFindIniMethods
403  *
404  * PARAMETERS:  ACPI_WALK_CALLBACK
405  *
406  * RETURN:      ACPI_STATUS
407  *
408  * DESCRIPTION: Called during namespace walk. Finds objects named _INI under
409  *              device/processor/thermal objects, and marks the entire subtree
410  *              with a SUBTREE_HAS_INI flag. This flag is used during the
411  *              subsequent device initialization walk to avoid entire subtrees
412  *              that do not contain an _INI.
413  *
414  *****************************************************************************/

416 static ACPI_STATUS
417 AcpiNsFindIniMethods (
418     ACPI_HANDLE             ObjHandle,
419     UINT32                  NestingLevel,
420     void                    *Context,
421     void                    **ReturnValue)
422 {
423     ACPI_DEVICE_WALK_INFO   *Info = ACPI_CAST_PTR (ACPI_DEVICE_WALK_INFO, Contex
424     ACPI_NAMESPACE_NODE     *Node;
425     ACPI_NAMESPACE_NODE     *ParentNode;


428     /* Keep count of device/processor/thermal objects */

430     Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
431     if ((Node->Type == ACPI_TYPE_DEVICE)    ||
432         (Node->Type == ACPI_TYPE_PROCESSOR) ||
433         (Node->Type == ACPI_TYPE_THERMAL))
434     {
435         Info->DeviceCount++;
436         return (AE_OK);
437     }

439     /* We are only looking for methods named _INI */

441     if (!ACPI_COMPARE_NAME (Node->Name.Ascii, METHOD_NAME__INI))
```



```
442     {
443         return (AE_OK);
444     }

446     /*
447      * The only _INI methods that we care about are those that are
448      * present under Device, Processor, and Thermal objects.
449      */
450     ParentNode = Node->Parent;
451     switch (ParentNode->Type)
452     {
453     case ACPI_TYPE_DEVICE:
454     case ACPI_TYPE_PROCESSOR:
455     case ACPI_TYPE_THERMAL:

457         /* Mark parent and bubble up the INI present flag to the root */

459         while (ParentNode)
460         {
461             ParentNode->Flags |= ANOBJ_SUBTREE_HAS_INI;
462             ParentNode = ParentNode->Parent;
463         }
464         break;

466     default:

468         break;
469     }

471     return (AE_OK);
472 }


475 /******************************************************************************
476  *
477  * FUNCTION:    AcpiNsInitOneDevice
478  *
479  * PARAMETERS:  ACPI_WALK_CALLBACK
480  *
481  * RETURN:      ACPI_STATUS
482  *
483  * DESCRIPTION: This is called once per device soon after ACPI is enabled
484  *              to initialize each device. It determines if the device is
485  *              present, and if so, calls _INI.
486  *
487  *****************************************************************************/

489 static ACPI_STATUS
490 AcpiNsInitOneDevice (
491     ACPI_HANDLE             ObjHandle,
492     UINT32                  NestingLevel,
493     void                    *Context,
494     void                    **ReturnValue)
495 {
496     ACPI_DEVICE_WALK_INFO   *WalkInfo = ACPI_CAST_PTR (ACPI_DEVICE_WALK_INFO, Co
497     ACPI_EVALUATE_INFO      *Info = WalkInfo->EvaluateInfo;
498     UINT32                  Flags;
499     ACPI_STATUS             Status;
500     ACPI_NAMESPACE_NODE     *DeviceNode;


503     ACPI_FUNCTION_TRACE (NsInitOneDevice);


506     /* We are interested in Devices, Processors and ThermalZones only */
```

```
508     DeviceNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
509     if ((DeviceNode->Type != ACPI_TYPE_DEVICE)    &&
510         (DeviceNode->Type != ACPI_TYPE_PROCESSOR) &&
511         (DeviceNode->Type != ACPI_TYPE_THERMAL))
512     {
513         return_ACPI_STATUS (AE_OK);
514     }
515
516     /*
517      * Because of an earlier namespace analysis, all subtrees that contain an
518      * _INI method are tagged.
519      *
520      * If this device subtree does not contain any _INI methods, we
521      * can exit now and stop traversing this entire subtree.
522      */
523     if (!(DeviceNode->Flags & ANOBJ_SUBTREE_HAS_INI))
524     {
525         return_ACPI_STATUS (AE_CTRL_DEPTH);
526     }
527
528     /*
529      * Run _STA to determine if this device is present and functioning. We
530      * must know this information for two important reasons (from ACPI spec):
531      *
532      * 1) We can only run _INI if the device is present.
533      * 2) We must abort the device tree walk on this subtree if the device is
534      *    not present and is not functional (we will not examine the children)
535      *
536      * The _STA method is not required to be present under the device, we
537      * assume the device is present if _STA does not exist.
538      */
539     ACPI_DEBUG_EXEC (AcpiUtDisplayInitPathname (
540         ACPI_TYPE_METHOD, DeviceNode, METHOD_NAME__STA));
541
542     Status = AcpiUtExecute_STA (DeviceNode, &Flags);
543     if (ACPI_FAILURE (Status))
544     {
545         /* Ignore error and move on to next device */
546
547         return_ACPI_STATUS (AE_OK);
548     }
549
550     /*
551      * Flags == -1 means that _STA was not found. In this case, we assume that
552      * the device is both present and functional.
553      *
554      * From the ACPI spec, description of _STA:
555      *
556      * "If a device object (including the processor object) does not have an
557      * _STA object, then OSPM assumes that all of the above bits are set (in
558      * other words, the device is present, ..., and functioning)"
559      */
560     if (Flags != ACPI_UINT32_MAX)
561     {
562         WalkInfo->Num_STA++;
563     }
564
565     /*
566      * Examine the PRESENT and FUNCTIONING status bits
567      *
568      * Note: ACPI spec does not seem to specify behavior for the present but
569      * not functioning case, so we assume functioning if present.
570      */
571     if (!(Flags & ACPI_STA_DEVICE_PRESENT))
572     {
573         /* Device is not present, we must examine the Functioning bit */
```



```
575         if (Flags & ACPI_STA_DEVICE_FUNCTIONING)
576         {
577             /*
578              * Device is not present but is "functioning". In this case,
579              * we will not run _INI, but we continue to examine the children
580              * of this device.
581              *
582              * From the ACPI spec, description of _STA: (Note - no mention
583              * of whether to run _INI or not on the device in question)
584              *
585              * "_STA may return bit 0 clear (not present) with bit 3 set
586              * (device is functional). This case is used to indicate a valid
587              * device for which no device driver should be loaded (for example,
588              * a bridge device.) Children of this device may be present and
589              * valid. OSPM should continue enumeration below a device whose
590              * _STA returns this bit combination"
591              */
592             return_ACPI_STATUS (AE_OK);
593         }
594         else
595         {
596             /*
597              * Device is not present and is not functioning. We must abort the
598              * walk of this subtree immediately -- don't look at the children
599              * of such a device.
600              *
601              * From the ACPI spec, description of _INI:
602              *
603              * "If the _STA method indicates that the device is not present,
604              * OSPM will not run the _INI and will not examine the children
605              * of the device for _INI methods"
606              */
607             return_ACPI_STATUS (AE_CTRL_DEPTH);
608         }
609     }
610
611     /*
612      * The device is present or is assumed present if no _STA exists.
613      * Run the _INI if it exists (not required to exist)
614      *
615      * Note: We know there is an _INI within this subtree, but it may not be
616      * under this particular device, it may be lower in the branch.
617      */
618     ACPI_DEBUG_EXEC (AcpiUtDisplayInitPathname (
619         ACPI_TYPE_METHOD, DeviceNode, METHOD_NAME__INI));
620
621     ACPI_MEMSET (Info, 0, sizeof (ACPI_EVALUATE_INFO));
622     Info->PrefixNode = DeviceNode;
623     Info->RelativePathname = METHOD_NAME__INI;
622     Info->Pathname = METHOD_NAME__INI;
624     Info->Parameters = NULL;
625     Info->Flags = ACPI_IGNORE_RETURN_VALUE;
626
627     Status = AcpiNsEvaluate (Info);
628     if (ACPI_SUCCESS (Status))
629     {
630         WalkInfo->Num_INI++;
631
632         if ((AcpiDbgLevel <= ACPI_LV_ALL_EXCEPTIONS) &&
633             (!(AcpiDbgLevel & ACPI_LV_INFO)))
634         {
631             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT, "."));
632         }
633     }
634
```

```
 633 #ifdef ACPI_DEBUG_OUTPUT
 634     else if (Status != AE_NOT_FOUND)
 635     {
 636         /* Ignore error and move on to next device */

 638         char *ScopeName = AcpiNsGetExternalPathname (Info->Node);
 643         char *ScopeName = AcpiNsGetExternalPathname (Info->ResolvedNode);

 640         ACPI_EXCEPTION ((AE_INFO, Status, "during %s._INI execution",
 641             ScopeName));
 642         ACPI_FREE (ScopeName);
 643     }
 644 #endif

 646     /* Ignore errors from above */

 648     Status = AE_OK;

 650     /*
 651      * The _INI method has been run if present; call the Global Initialization
 652      * Handler for this device.
 653      */
 654     if (AcpiGbl_InitHandler)
 655     {
 656         Status = AcpiGbl_InitHandler (DeviceNode, ACPI_INIT_DEVICE_INI);
 657     }

 659     return_ACPI_STATUS (Status);
 660 }
```

_____unchanged_portion_omitted_

```
**********************************************************
   10010 Thu Dec 26 13:49:18 2013
new/usr/src/common/acpica/components/namespace/nsload.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: nsload - namespace loading/expanding/contracting procedures
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSLOAD_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acdispat.h"
#include "actables.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsload")

/* Local prototypes */

#ifdef ACPI_FUTURE_IMPLEMENTATION
ACPI_STATUS
```



```
AcpiNsUnloadNamespace (
    ACPI_HANDLE             Handle);

static ACPI_STATUS
AcpiNsDeleteSubtree (
    ACPI_HANDLE             StartHandle);
#endif


#ifndef ACPI_NO_METHOD_EXECUTION
/******************************************************************************
 *
 * FUNCTION:    AcpiNsLoadTable
 *
 * PARAMETERS:  TableIndex      - Index for table to be loaded
 *              Node            - Owning NS node
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Load one ACPI table into the namespace
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsLoadTable (
    UINT32                  TableIndex,
    ACPI_NAMESPACE_NODE     *Node)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (NsLoadTable);


    /*
     * Parse the table and load the namespace with all named
     * objects found within. Control methods are NOT parsed
     * at this time. In fact, the control methods cannot be
     * parsed until the entire namespace is loaded, because
     * if a control method makes a forward reference (call)
     * to another control method, we can't continue parsing
     * because we don't know how many arguments to parse next!
     */
    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* If table already loaded into namespace, just return */

    if (AcpiTbIsTableLoaded (TableIndex))
    {
        Status = AE_ALREADY_EXISTS;
        goto Unlock;
    }

    ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
        "**** Loading table into namespace ****\n"));

    Status = AcpiTbAllocateOwnerId (TableIndex);
    if (ACPI_FAILURE (Status))
    {
        goto Unlock;
    }
```

```
126     Status = AcpiNsParseTable (TableIndex, Node);
127     if (ACPI_SUCCESS (Status))
128     {
129         AcpiTbSetTableLoadedFlag (TableIndex, TRUE);
130     }
131     else
132     {
133         (void) AcpiTbReleaseOwnerId (TableIndex);
134     }

136 Unlock:
137     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

139     if (ACPI_FAILURE (Status))
140     {
141         return_ACPI_STATUS (Status);
142     }

144     /*
145      * Now we can parse the control methods. We always parse
146      * them here for a sanity check, and if configured for
147      * just-in-time parsing, we delete the control method
148      * parse trees.
149      */
150     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
151         "**** Begin Table Method Parsing and Object Initialization\n"));

153     Status = AcpiDsInitializeObjects (TableIndex, Node);

155     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
156         "**** Completed Table Method Parsing and Object Initialization\n"));

158     return_ACPI_STATUS (Status);
159 }
```

unchanged_portion_omitted_

```
354 #endif
355 #endif
```

```
**********************************************************
    8693 Thu Dec 26 13:49:19 2013
new/usr/src/common/acpica/components/namespace/nsnames.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: nsnames - Name manipulation and search
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSNAMES_C__

#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsnames")


/******************************************************************************
 *
 * FUNCTION:    AcpiNsBuildExternalPath
 *
```



```
 * PARAMETERS:  Node            - NS node whose pathname is needed
 *              Size            - Size of the pathname
 *              *NameBuffer     - Where to return the pathname
 *
 * RETURN:      Status
 *              Places the pathname into the NameBuffer, in external format
 *              (name segments separated by path separators)
 *
 * DESCRIPTION: Generate a full pathaname
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsBuildExternalPath (
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_SIZE               Size,
    char                    *NameBuffer)
{
    ACPI_SIZE               Index;
    ACPI_NAMESPACE_NODE     *ParentNode;


    ACPI_FUNCTION_ENTRY ();


    /* Special case for root */

    Index = Size - 1;
    if (Index < ACPI_NAME_SIZE)
    {
        NameBuffer[0] = AML_ROOT_PREFIX;
        NameBuffer[1] = 0;
        return (AE_OK);
    }

    /* Store terminator byte, then build name backwards */

    ParentNode = Node;
    NameBuffer[Index] = 0;

    while ((Index > ACPI_NAME_SIZE) && (ParentNode != AcpiGbl_RootNode))
    {
        Index -= ACPI_NAME_SIZE;

        /* Put the name into the buffer */

        ACPI_MOVE_32_TO_32 ((NameBuffer + Index), &ParentNode->Name);
        ParentNode = ParentNode->Parent;

        /* Prefix name with the path separator */

        Index--;
        NameBuffer[Index] = ACPI_PATH_SEPARATOR;
    }

    /* Overwrite final separator with the root prefix character */

    NameBuffer[Index] = AML_ROOT_PREFIX;

    if (Index != 0)
    {
        ACPI_ERROR ((AE_INFO,
            "Could not construct external pathname; index=%u, size=%u, Path=%s",
            (UINT32) Index, (UINT32) Size, &NameBuffer[Size]));

        return (AE_BAD_PARAMETER);
```

```
 126     }

 128     return (AE_OK);
 129 }
______unchanged_portion_omitted_


 189 /******************************************************************************
 190  *
 191  * FUNCTION:    AcpiNsGetPathnameLength
 192  *
 193  * PARAMETERS:  Node        - Namespace node
 194  *
 195  * RETURN:      Length of path, including prefix
 196  *
 197  * DESCRIPTION: Get the length of the pathname string for this node
 198  *
 199  *****************************************************************************/

 201 ACPI_SIZE
 202 AcpiNsGetPathnameLength (
 203     ACPI_NAMESPACE_NODE     *Node)
 204 {
 205     ACPI_SIZE               Size;
 206     ACPI_NAMESPACE_NODE     *NextNode;


 209     ACPI_FUNCTION_ENTRY ();


 212     /*
 213      * Compute length of pathname as 5 * number of name segments.
 214      * Go back up the parent tree to the root
 215      */
 216     Size = 0;
 217     NextNode = Node;

 219     while (NextNode && (NextNode != AcpiGbl_RootNode))
 220     {
 221         if (ACPI_GET_DESCRIPTOR_TYPE (NextNode) != ACPI_DESC_TYPE_NAMED)
 222         {
 223             ACPI_ERROR ((AE_INFO,
 224                 "Invalid Namespace Node (%p) while traversing namespace",
 225                 NextNode));
 226             return (0);
 226             return 0;
 227         }
 228         Size += ACPI_PATH_SEGMENT_LENGTH;
 229         NextNode = NextNode->Parent;
 230     }

 232     if (!Size)
 233     {
 234         Size = 1; /* Root node case */
 235     }

 237     return (Size + 1);  /* +1 for null string terminator */
 238 }
______unchanged_portion_omitted_
```

****
13934 Thu Dec 26 13:49:19 2013
new/usr/src/common/acpica/components/namespace/nsobject.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: nsobject - Utilities for objects attached to namespace
 *                         table entries
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSOBJECT_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsobject")


/******************************************************************************
 *
 * FUNCTION:    AcpiNsAttachObject
```



```
 *
 * PARAMETERS:  Node                - Parent Node
 *              Object              - Object to be attached
 *              Type                - Type of object, or ACPI_TYPE_ANY if not
 *                                    known
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Record the given object as the value associated with the
 *              name whose ACPI_HANDLE is passed. If Object is NULL
 *              and Type is ACPI_TYPE_ANY, set the name as having no value.
 *              Note: Future may require that the Node->Flags field be passed
 *              as a parameter.
 *
 * MUTEX:       Assumes namespace is locked
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsAttachObject (
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_OPERAND_OBJECT     *Object,
    ACPI_OBJECT_TYPE        Type)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_OPERAND_OBJECT     *LastObjDesc;
    ACPI_OBJECT_TYPE        ObjectType = ACPI_TYPE_ANY;


    ACPI_FUNCTION_TRACE (NsAttachObject);


    /*
     * Parameter validation
     */
    if (!Node)
    {
        /* Invalid handle */

        ACPI_ERROR ((AE_INFO, "Null NamedObj handle"));
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (!Object && (ACPI_TYPE_ANY != Type))
    {
        /* Null object */

        ACPI_ERROR ((AE_INFO,
            "Null object, but type not ACPI_TYPE_ANY"));
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (ACPI_GET_DESCRIPTOR_TYPE (Node) != ACPI_DESC_TYPE_NAMED)
    {
        /* Not a name handle */

        ACPI_ERROR ((AE_INFO, "Invalid handle %p [%s]",
            Node, AcpiUtGetDescriptorName (Node)));
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    /* Check if this object is already attached */

    if (Node->Object == Object)
    {
        ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
```

```
126             "Obj %p already installed in NameObj %p\n",
127             Object, Node));

129         return_ACPI_STATUS (AE_OK);
130     }

132     /* If null object, we will just install it */

134     if (!Object)
135     {
136         ObjDesc    = NULL;
137         ObjectType = ACPI_TYPE_ANY;
138     }

140     /*
141      * If the source object is a namespace Node with an attached object,
142      * we will use that (attached) object
143      */
144     else if ((ACPI_GET_DESCRIPTOR_TYPE (Object) == ACPI_DESC_TYPE_NAMED) &&
145             ((ACPI_NAMESPACE_NODE *) Object)->Object)
146     {
147         /*
148          * Value passed is a name handle and that name has a
149          * non-null value. Use that name's value and type.
150          */
151         ObjDesc    = ((ACPI_NAMESPACE_NODE *) Object)->Object;
152         ObjectType = ((ACPI_NAMESPACE_NODE *) Object)->Type;
153     }

155     /*
156      * Otherwise, we will use the parameter object, but we must type
157      * it first
158      */
159     else
160     {
161         ObjDesc = (ACPI_OPERAND_OBJECT  *) Object;

163         /* Use the given type */

165         ObjectType = Type;
166     }

168     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Installing %p into Node %p [%4.4s]\n",
169         ObjDesc, Node, AcpiUtGetNodeName (Node)));

171     /* Detach an existing attached object if present */

173     if (Node->Object)
174     {
175         AcpiNsDetachObject (Node);
176     }

178     if (ObjDesc)
179     {
180         /*
181          * Must increment the new value's reference count
182          * (if it is an internal object)
183          */
184         AcpiUtAddReference (ObjDesc);

186         /*
187          * Handle objects with multiple descriptors - walk
188          * to the end of the descriptor list
189          */
190         LastObjDesc = ObjDesc;
191         while (LastObjDesc->Common.NextObject)
```



```
192         {
193             LastObjDesc = LastObjDesc->Common.NextObject;
194         }

196         /* Install the object at the front of the object list */

198         LastObjDesc->Common.NextObject = Node->Object;
199     }

201     Node->Type     = (UINT8) ObjectType;
202     Node->Object   = ObjDesc;

204     return_ACPI_STATUS (AE_OK);
205 }
```

_____unchanged_portion_omitted_

****
7053 Thu Dec 26 13:49:19 2013
new/usr/src/common/acpica/components/namespace/nsparse.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: nsparse - namespace interface to AML parser
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSPARSE_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acparser.h"
#include "acdispat.h"
#include "actables.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsparse")


/*******************************************************************************
 *
```



```
 * FUNCTION:    NsOneCompleteParse
 *
 * PARAMETERS:  PassNumber              - 1 or 2
 *              TableDesc               - The table to be parsed.
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform one complete parse of an ACPI/AML table.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiNsOneCompleteParse (
    UINT32                  PassNumber,
    UINT32                  TableIndex,
    ACPI_NAMESPACE_NODE     *StartNode)
{
    ACPI_PARSE_OBJECT       *ParseRoot;
    ACPI_STATUS             Status;
    UINT32                  AmlLength;
    UINT8                   *AmlStart;
    ACPI_WALK_STATE         *WalkState;
    ACPI_TABLE_HEADER       *Table;
    ACPI_OWNER_ID           OwnerId;


    ACPI_FUNCTION_TRACE (NsOneCompleteParse);


    Status = AcpiTbGetOwnerId (TableIndex, &OwnerId);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Create and init a Root Node */

    ParseRoot = AcpiPsCreateScopeOp ();
    if (!ParseRoot)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    /* Create and initialize a new walk state */

    WalkState = AcpiDsCreateWalkState (OwnerId, NULL, NULL, NULL);
    if (!WalkState)
    {
        AcpiPsFreeOp (ParseRoot);
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    Status = AcpiGetTableByIndex (TableIndex, &Table);
    if (ACPI_FAILURE (Status))
    {
        AcpiDsDeleteWalkState (WalkState);
        AcpiPsFreeOp (ParseRoot);
        return_ACPI_STATUS (Status);
    }

    /* Table must consist of at least a complete header */

    if (Table->Length < sizeof (ACPI_TABLE_HEADER))
    {
        Status = AE_BAD_HEADER;
    }
```

```
126     else
127     {
128         AmlStart = (UINT8 *) Table + sizeof (ACPI_TABLE_HEADER);
129         AmlLength = Table->Length - sizeof (ACPI_TABLE_HEADER);
130         Status = AcpiDsInitAmlWalk (WalkState, ParseRoot, NULL,
131                     AmlStart, AmlLength, NULL, (UINT8) PassNumber);
132     }

134     if (ACPI_FAILURE (Status))
135     {
136         AcpiDsDeleteWalkState (WalkState);
137         goto Cleanup;
138     }

140     /* StartNode is the default location to load the table  */

142     if (StartNode && StartNode != AcpiGbl_RootNode)
143     {
144         Status = AcpiDsScopeStackPush (StartNode, ACPI_TYPE_METHOD, WalkState);
145         if (ACPI_FAILURE (Status))
146         {
147             AcpiDsDeleteWalkState (WalkState);
148             goto Cleanup;
149         }
150     }

152     /* Parse the AML */

154     ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "*PARSE* pass %u parse\n", PassNumber));
155     Status = AcpiPsParseAml (WalkState);

157 Cleanup:
158     AcpiPsDeleteParseTree (ParseRoot);
159     return_ACPI_STATUS (Status);
160 }
```

unchanged_portion_omitted_

```
**********************************************************
   14310 Thu Dec 26 13:49:20 2013
new/usr/src/common/acpica/components/namespace/nspredef.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nspredef - Validation of ACPI predefined methods and objects
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define ACPI_CREATE_PREDEFINED_TABLE

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acpredef.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nspredef")


/******************************************************************************
 *
 * This module validates predefined ACPI objects that appear in the namespace,
 * at the time they are evaluated (via AcpiEvaluateObject). The purpose of this
 * validation is to detect problems with BIOS-exposed predefined ACPI objects
```



```
 * before the results are returned to the ACPI-related drivers.
 *
 * There are several areas that are validated:
 *
 *  1) The number of input arguments as defined by the method/object in the
 *     ASL is validated against the ACPI specification.
 *  2) The type of the return object (if any) is validated against the ACPI
 *     specification.
 *  3) For returned package objects, the count of package elements is
 *     validated, as well as the type of each package element. Nested
 *     packages are supported.
 *
 * For any problems found, a warning message is issued.
 *
 *****************************************************************************/


/* Local prototypes */

static ACPI_STATUS
AcpiNsCheckReference (
    ACPI_EVALUATE_INFO          *Info,
    ACPI_OPERAND_OBJECT         *ReturnObject);

static UINT32
AcpiNsGetBitmappedType (
    ACPI_OPERAND_OBJECT         *ReturnObject);


/******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckReturnValue
 *
 * PARAMETERS:  Node            - Namespace node for the method/object
 *              Info            - Method execution information block
 *              UserParamCount  - Number of parameters actually passed
 *              ReturnStatus    - Status from the object evaluation
 *              ReturnObjectPtr - Pointer to the object returned from the
 *                                evaluation of a method or object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check the value returned from a predefined name.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsCheckReturnValue (
    ACPI_NAMESPACE_NODE         *Node,
    ACPI_EVALUATE_INFO          *Info,
    UINT32                      UserParamCount,
    ACPI_STATUS                 ReturnStatus,
    ACPI_OPERAND_OBJECT         **ReturnObjectPtr)
{
    ACPI_STATUS                 Status;
    const ACPI_PREDEFINED_INFO  *Predefined;


    /* If not a predefined name, we cannot validate the return object */

    Predefined = Info->Predefined;
    if (!Predefined)
    {
        return (AE_OK);
    }
```

```
    /*
     * If the method failed or did not actually return an object, we cannot
     * validate the return object
     */
    if ((ReturnStatus != AE_OK) &&
        (ReturnStatus != AE_CTRL_RETURN_VALUE))
    {
        return (AE_OK);
    }

    /*
     * Return value validation and possible repair.
     *
     * 1) Don't perform return value validation/repair if this feature
     * has been disabled via a global option.
     *
     * 2) We have a return value, but if one wasn't expected, just exit,
     * this is not a problem. For example, if the "Implicit Return"
     * feature is enabled, methods will always return a value.
     *
     * 3) If the return value can be of any type, then we cannot perform
     * any validation, just exit.
     */
    if (AcpiGbl_DisableAutoRepair ||
        (!Predefined->Info.ExpectedBtypes) ||
        (Predefined->Info.ExpectedBtypes == ACPI_RTYPE_ALL))
    {
        return (AE_OK);
    }

    /*
     * Check that the type of the main return object is what is expected
     * for this predefined name
     */
    Status = AcpiNsCheckObjectType (Info, ReturnObjectPtr,
        Predefined->Info.ExpectedBtypes, ACPI_NOT_PACKAGE_ELEMENT);
    if (ACPI_FAILURE (Status))
    {
        goto Exit;
    }

    /*
     *
     * 4) If there is no return value and it is optional, just return
     * AE_OK (_WAK).
     */
    if (!(*ReturnObjectPtr))
    {
        goto Exit;
    }

    /*
     * For returned Package objects, check the type of all sub-objects.
     * Note: Package may have been newly created by call above.
     */
    if ((*ReturnObjectPtr)->Common.Type == ACPI_TYPE_PACKAGE)
    {
        Info->ParentPackage = *ReturnObjectPtr;
        Status = AcpiNsCheckPackage (Info, ReturnObjectPtr);
        if (ACPI_FAILURE (Status))
        {
            /* We might be able to fix some errors */

            if ((Status != AE_AML_OPERAND_TYPE) &&
                (Status != AE_AML_OPERAND_VALUE))
            {
```



```
                goto Exit;
            }
        }
    }

    /*
     * The return object was OK, or it was successfully repaired above.
     * Now make some additional checks such as verifying that package
     * objects are sorted correctly (if required) or buffer objects have
     * the correct data width (bytes vs. dwords). These repairs are
     * performed on a per-name basis, i.e., the code is specific to
     * particular predefined names.
     */
    Status = AcpiNsComplexRepairs (Info, Node, Status, ReturnObjectPtr);

Exit:
    /*
     * If the object validation failed or if we successfully repaired one
     * or more objects, mark the parent node to suppress further warning
     * messages during the next evaluation of the same method/object.
     */
    if (ACPI_FAILURE (Status) ||
       (Info->ReturnFlags & ACPI_OBJECT_REPAIRED))
    {
        Node->Flags |= ANOBJ_EVALUATED;
    }

    return (Status);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckObjectType
 *
 * PARAMETERS:  Info            - Method execution information block
 *              ReturnObjectPtr - Pointer to the object returned from the
 *                                evaluation of a method or object
 *              ExpectedBtypes  - Bitmap of expected return type(s)
 *              PackageIndex    - Index of object within parent package (if
 *                                applicable - ACPI_NOT_PACKAGE_ELEMENT
 *                                otherwise)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check the type of the return object against the expected object
 *              type(s). Use of Btype allows multiple expected object types.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiNsCheckObjectType (
    ACPI_EVALUATE_INFO          *Info,
    ACPI_OPERAND_OBJECT         **ReturnObjectPtr,
    UINT32                      ExpectedBtypes,
    UINT32                      PackageIndex)
{
    ACPI_OPERAND_OBJECT         *ReturnObject = *ReturnObjectPtr;
    ACPI_STATUS                 Status = AE_OK;
    char                        TypeBuffer[48]; /* Room for 5 types */


    /* A Namespace node should not get here, but make sure */

    if (ReturnObject &&
        ACPI_GET_DESCRIPTOR_TYPE (ReturnObject) == ACPI_DESC_TYPE_NAMED)
```

```
259     {
260         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
261             "Invalid return type - Found a Namespace node [%4.4s] type %s",
262             ReturnObject->Node.Name.Ascii,
263             AcpiUtGetTypeName (ReturnObject->Node.Type)));
264         return (AE_AML_OPERAND_TYPE);
265     }

267     /*
268      * Convert the object type (ACPI_TYPE_xxx) to a bitmapped object type.
269      * The bitmapped type allows multiple possible return types.
270      *
271      * Note, the cases below must handle all of the possible types returned
272      * from all of the predefined names (including elements of returned
273      * packages)
274      */
275     Info->ReturnBtype = AcpiNsGetBitmappedType (ReturnObject);
276     if (Info->ReturnBtype == ACPI_RTYPE_ANY)
277     {
278         /* Not one of the supported objects, must be incorrect */
279         goto TypeErrorExit;
280     }

282     /* For reference objects, check that the reference type is correct */

284     if ((Info->ReturnBtype & ExpectedBtypes) == ACPI_RTYPE_REFERENCE)
285     {
286         Status = AcpiNsCheckReference (Info, ReturnObject);
287         return (Status);
288     }

290     /* Attempt simple repair of the returned object if necessary */

292     Status = AcpiNsSimpleRepair (Info, ExpectedBtypes,
293         PackageIndex, ReturnObjectPtr);
294     if (ACPI_SUCCESS (Status))
295     {
296         return (AE_OK); /* Successful repair */
297     }


300 TypeErrorExit:

302     /* Create a string with all expected types for this predefined object */

304     AcpiUtGetExpectedReturnTypes (TypeBuffer, ExpectedBtypes);

306     if (!ReturnObject)
307     {
308         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
309             "Expected return object of type %s",
310             TypeBuffer));
311     }
312     else if (PackageIndex == ACPI_NOT_PACKAGE_ELEMENT)
313     {
314         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
315             "Return type mismatch - found %s, expected %s",
316             AcpiUtGetObjectTypeName (ReturnObject), TypeBuffer));
317     }
318     else
319     {
320         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
321             "Return Package type mismatch at index %u - "
322             "found %s, expected %s", PackageIndex,
323             AcpiUtGetObjectTypeName (ReturnObject), TypeBuffer));
324     }
```



```
326     return (AE_AML_OPERAND_TYPE);
327 }


330 /******************************************************************************
331  *
332  * FUNCTION:    AcpiNsCheckReference
333  *
334  * PARAMETERS:  Info            - Method execution information block
335  *              ReturnObject    - Object returned from the evaluation of a
336  *                                method or object
337  *
338  * RETURN:      Status
339  *
340  * DESCRIPTION: Check a returned reference object for the correct reference
341  *              type. The only reference type that can be returned from a
342  *              predefined method is a named reference. All others are invalid.
343  *
344  *****************************************************************************/

346 static ACPI_STATUS
347 AcpiNsCheckReference (
348     ACPI_EVALUATE_INFO          *Info,
349     ACPI_OPERAND_OBJECT         *ReturnObject)
350 {

352     /*
353      * Check the reference object for the correct reference type (opcode).
354      * The only type of reference that can be converted to an ACPI_OBJECT is
355      * a reference to a named object (reference class: NAME)
356      */
357     if (ReturnObject->Reference.Class == ACPI_REFCLASS_NAME)
358     {
359         return (AE_OK);
360     }

362     ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
363         "Return type mismatch - unexpected reference object type [%s] %2.2X",
364         AcpiUtGetReferenceName (ReturnObject),
365         ReturnObject->Reference.Class));

367     return (AE_AML_OPERAND_TYPE);
368 }


371 /******************************************************************************
372  *
373  * FUNCTION:    AcpiNsGetBitmappedType
374  *
375  * PARAMETERS:  ReturnObject    - Object returned from method/obj evaluation
376  *
377  * RETURN:      Object return type. ACPI_RTYPE_ANY indicates that the object
378  *              type is not supported. ACPI_RTYPE_NONE indicates that no
379  *              object was returned (ReturnObject is NULL).
380  *
381  * DESCRIPTION: Convert object type into a bitmapped object return type.
382  *
383  *****************************************************************************/

385 static UINT32
386 AcpiNsGetBitmappedType (
387     ACPI_OPERAND_OBJECT         *ReturnObject)
388 {
389     UINT32                      ReturnBtype;
```

```
392     if (!ReturnObject)
393     {
394         return (ACPI_RTYPE_NONE);
395     }

397     /* Map ACPI_OBJECT_TYPE to internal bitmapped type */

399     switch (ReturnObject->Common.Type)
400     {
401     case ACPI_TYPE_INTEGER:

403         ReturnBtype = ACPI_RTYPE_INTEGER;
404         break;

406     case ACPI_TYPE_BUFFER:

408         ReturnBtype = ACPI_RTYPE_BUFFER;
409         break;

411     case ACPI_TYPE_STRING:

413         ReturnBtype = ACPI_RTYPE_STRING;
414         break;

416     case ACPI_TYPE_PACKAGE:

418         ReturnBtype = ACPI_RTYPE_PACKAGE;
419         break;

421     case ACPI_TYPE_LOCAL_REFERENCE:

423         ReturnBtype = ACPI_RTYPE_REFERENCE;
424         break;

426     default:

428         /* Not one of the supported objects, must be incorrect */

430         ReturnBtype = ACPI_RTYPE_ANY;
431         break;
432     }

434     return (ReturnBtype);
435 }
```

```
**********************************************************
   20409 Thu Dec 26 13:49:20 2013
new/usr/src/common/acpica/components/namespace/nsprepkg.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsprepkg - Validation of package objects for predefined names
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acpredef.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsprepkg")


/* Local prototypes */

static ACPI_STATUS
AcpiNsCheckPackageList (
    ACPI_EVALUATE_INFO          *Info,
    const ACPI_PREDEFINED_INFO  *Package,
    ACPI_OPERAND_OBJECT         **Elements,
    UINT32                      Count);
```



```
static ACPI_STATUS
AcpiNsCheckPackageElements (
    ACPI_EVALUATE_INFO          *Info,
    ACPI_OPERAND_OBJECT         **Elements,
    UINT8                       Type1,
    UINT32                      Count1,
    UINT8                       Type2,
    UINT32                      Count2,
    UINT32                      StartIndex);


/******************************************************************************
 *
 * FUNCTION:    AcpiNsCheckPackage
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Check a returned package object for the correct count and
 *              correct type of all sub-objects.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsCheckPackage (
    ACPI_EVALUATE_INFO          *Info,
    ACPI_OPERAND_OBJECT         **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT         *ReturnObject = *ReturnObjectPtr;
    const ACPI_PREDEFINED_INFO  *Package;
    ACPI_OPERAND_OBJECT         **Elements;
    ACPI_STATUS                 Status = AE_OK;
    UINT32                      ExpectedCount;
    UINT32                      Count;
    UINT32                      i;


    ACPI_FUNCTION_NAME (NsCheckPackage);


    /* The package info for this name is in the next table entry */

    Package = Info->Predefined + 1;

    ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
        "%s Validating return Package of Type %X, Count %X\n",
        Info->FullPathname, Package->RetInfo.Type,
        ReturnObject->Package.Count));

    /*
     * For variable-length Packages, we can safely remove all embedded
     * and trailing NULL package elements
     */
    AcpiNsRemoveNullElements (Info, Package->RetInfo.Type, ReturnObject);

    /* Extract package count and elements array */

    Elements = ReturnObject->Package.Elements;
    Count = ReturnObject->Package.Count;

    /*
     * Most packages must have at least one element. The only exception
```

```
128      * is the variable-length package (ACPI_PTYPE1_VAR).
129      */
130     if (!Count)
131     {
132         if (Package->RetInfo.Type == ACPI_PTYPE1_VAR)
133         {
134             return (AE_OK);
135         }
136
137         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
138             "Return Package has no elements (empty)"));
139
140         return (AE_AML_OPERAND_VALUE);
141     }
142
143     /*
144      * Decode the type of the expected package contents
145      *
146      * PTYPE1 packages contain no subpackages
147      * PTYPE2 packages contain sub-packages
148      */
149     switch (Package->RetInfo.Type)
150     {
151     case ACPI_PTYPE1_FIXED:
152         /*
153          * The package count is fixed and there are no sub-packages
154          *
155          * If package is too small, exit.
156          * If package is larger than expected, issue warning but continue
157          */
158         ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
159         if (Count < ExpectedCount)
160         {
161             goto PackageTooSmall;
162         }
163         else if (Count > ExpectedCount)
164         {
165             ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
166                 "%s: Return Package is larger than needed - "
167                 "found %u, expected %u\n",
168                 Info->FullPathname, Count, ExpectedCount));
169         }
170
171         /* Validate all elements of the returned package */
172
173         Status = AcpiNsCheckPackageElements (Info, Elements,
174                     Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
175                     Package->RetInfo.ObjectType2, Package->RetInfo.Count2, 0);
176         break;
177
178     case ACPI_PTYPE1_VAR:
179         /*
180          * The package count is variable, there are no sub-packages, and all
181          * elements must be of the same type
182          */
183         for (i = 0; i < Count; i++)
184         {
185             Status = AcpiNsCheckObjectType (Info, Elements,
186                         Package->RetInfo.ObjectType1, i);
187             if (ACPI_FAILURE (Status))
188             {
189                 return (Status);
190             }
191             Elements++;
192         }
193         break;
```



```
195     case ACPI_PTYPE1_OPTION:
196         /*
197          * The package count is variable, there are no sub-packages. There are
198          * a fixed number of required elements, and a variable number of
199          * optional elements.
200          *
201          * Check if package is at least as large as the minimum required
202          */
203         ExpectedCount = Package->RetInfo3.Count;
204         if (Count < ExpectedCount)
205         {
206             goto PackageTooSmall;
207         }
208
209         /* Variable number of sub-objects */
210
211         for (i = 0; i < Count; i++)
212         {
213             if (i < Package->RetInfo3.Count)
214             {
215                 /* These are the required package elements (0, 1, or 2) */
216
217                 Status = AcpiNsCheckObjectType (Info, Elements,
218                             Package->RetInfo3.ObjectType[i], i);
219                 if (ACPI_FAILURE (Status))
220                 {
221                     return (Status);
222                 }
223             }
224             else
225             {
226                 /* These are the optional package elements */
227
228                 Status = AcpiNsCheckObjectType (Info, Elements,
229                             Package->RetInfo3.TailObjectType, i);
230                 if (ACPI_FAILURE (Status))
231                 {
232                     return (Status);
233                 }
234             }
235             Elements++;
236         }
237         break;
238
239     case ACPI_PTYPE2_REV_FIXED:
240
241         /* First element is the (Integer) revision */
242
243         Status = AcpiNsCheckObjectType (Info, Elements,
244                     ACPI_RTYPE_INTEGER, 0);
245         if (ACPI_FAILURE (Status))
246         {
247             return (Status);
248         }
249
250         Elements++;
251         Count--;
252
253         /* Examine the sub-packages */
254
255         Status = AcpiNsCheckPackageList (Info, Package, Elements, Count);
256         break;
257
258     case ACPI_PTYPE2_PKG_COUNT:
```

```
260         /* First element is the (Integer) count of sub-packages to follow */

262         Status = AcpiNsCheckObjectType (Info, Elements,
263                     ACPI_RTYPE_INTEGER, 0);
264         if (ACPI_FAILURE (Status))
265         {
266             return (Status);
267         }

269         /*
270          * Count cannot be larger than the parent package length, but allow it
271          * to be smaller. The >= accounts for the Integer above.
272          */
273         ExpectedCount = (UINT32) (*Elements)->Integer.Value;
274         if (ExpectedCount >= Count)
275         {
276             goto PackageTooSmall;
277         }

279         Count = ExpectedCount;
280         Elements++;

282         /* Examine the sub-packages */

284         Status = AcpiNsCheckPackageList (Info, Package, Elements, Count);
285         break;

287     case ACPI_PTYPE2:
288     case ACPI_PTYPE2_FIXED:
289     case ACPI_PTYPE2_MIN:
290     case ACPI_PTYPE2_COUNT:
291     case ACPI_PTYPE2_FIX_VAR:
292         /*
293          * These types all return a single Package that consists of a
294          * variable number of sub-Packages.
295          *
296          * First, ensure that the first element is a sub-Package. If not,
297          * the BIOS may have incorrectly returned the object as a single
298          * package instead of a Package of Packages (a common error if
299          * there is only one entry). We may be able to repair this by
300          * wrapping the returned Package with a new outer Package.
301          */
302         if (*Elements && ((*Elements)->Common.Type != ACPI_TYPE_PACKAGE))
303         {
304             /* Create the new outer package and populate it */

306             Status = AcpiNsWrapWithPackage (Info, ReturnObject, ReturnObjectPtr)
307             if (ACPI_FAILURE (Status))
308             {
309                 return (Status);
310             }

312             /* Update locals to point to the new package (of 1 element) */

314             ReturnObject = *ReturnObjectPtr;
315             Elements = ReturnObject->Package.Elements;
316             Count = 1;
317         }

319         /* Examine the sub-packages */

321         Status = AcpiNsCheckPackageList (Info, Package, Elements, Count);
322         break;

324     default:
```



```
326         /* Should not get here if predefined info table is correct */

328         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
329             "Invalid internal return type in table entry: %X",
330             Package->RetInfo.Type));

332         return (AE_AML_INTERNAL);
333     }

335     return (Status);


338 PackageTooSmall:

340     /* Error exit for the case with an incorrect package count */

342     ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
343         "Return Package is too small - found %u elements, expected %u",
344         Count, ExpectedCount));

346     return (AE_AML_OPERAND_VALUE);
347 }


350 /******************************************************************************
351  *
352  * FUNCTION:    AcpiNsCheckPackageList
353  *
354  * PARAMETERS:  Info            - Method execution information block
355  *              Package         - Pointer to package-specific info for method
356  *              Elements        - Element list of parent package. All elements
357  *                                of this list should be of type Package.
358  *              Count           - Count of subpackages
359  *
360  * RETURN:      Status
361  *
362  * DESCRIPTION: Examine a list of subpackages
363  *
364  *****************************************************************************/

366 static ACPI_STATUS
367 AcpiNsCheckPackageList (
368     ACPI_EVALUATE_INFO          *Info,
369     const ACPI_PREDEFINED_INFO  *Package,
370     ACPI_OPERAND_OBJECT         **Elements,
371     UINT32                      Count)
372 {
373     ACPI_OPERAND_OBJECT         *SubPackage;
374     ACPI_OPERAND_OBJECT         **SubElements;
375     ACPI_STATUS                 Status;
376     UINT32                      ExpectedCount;
377     UINT32                      i;
378     UINT32                      j;


381     /*
382      * Validate each sub-Package in the parent Package
383      *
384      * NOTE: assumes list of sub-packages contains no NULL elements.
385      * Any NULL elements should have been removed by earlier call
386      * to AcpiNsRemoveNullElements.
387      */
388     for (i = 0; i < Count; i++)
389     {
390         SubPackage = *Elements;
391         SubElements = SubPackage->Package.Elements;
```

```
392         Info->ParentPackage = SubPackage;
394         /* Each sub-object must be of type Package */
396         Status = AcpiNsCheckObjectType (Info, &SubPackage,
397                     ACPI_RTYPE_PACKAGE, i);
398         if (ACPI_FAILURE (Status))
399         {
400             return (Status);
401         }
403         /* Examine the different types of expected sub-packages */
405         Info->ParentPackage = SubPackage;
406         switch (Package->RetInfo.Type)
407         {
408         case ACPI_PTYPE2:
409         case ACPI_PTYPE2_PKG_COUNT:
410         case ACPI_PTYPE2_REV_FIXED:
412             /* Each subpackage has a fixed number of elements */
414             ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
415             if (SubPackage->Package.Count < ExpectedCount)
416             {
417                 goto PackageTooSmall;
418             }
420             Status = AcpiNsCheckPackageElements (Info, SubElements,
421                         Package->RetInfo.ObjectType1,
422                         Package->RetInfo.Count1,
423                         Package->RetInfo.ObjectType2,
424                         Package->RetInfo.Count2, 0);
425             if (ACPI_FAILURE (Status))
426             {
427                 return (Status);
428             }
429             break;
431         case ACPI_PTYPE2_FIX_VAR:
432             /*
433              * Each subpackage has a fixed number of elements and an
434              * optional element
435              */
436             ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
437             if (SubPackage->Package.Count < ExpectedCount)
438             {
439                 goto PackageTooSmall;
440             }
442             Status = AcpiNsCheckPackageElements (Info, SubElements,
443                         Package->RetInfo.ObjectType1,
444                         Package->RetInfo.Count1,
445                         Package->RetInfo.ObjectType2,
446                         SubPackage->Package.Count - Package->RetInfo.Count1, 0);
447             if (ACPI_FAILURE (Status))
448             {
449                 return (Status);
450             }
451             break;
453         case ACPI_PTYPE2_FIXED:
455             /* Each sub-package has a fixed length */
457             ExpectedCount = Package->RetInfo2.Count;
```



```
458             if (SubPackage->Package.Count < ExpectedCount)
459             {
460                 goto PackageTooSmall;
461             }
463             /* Check the type of each sub-package element */
465             for (j = 0; j < ExpectedCount; j++)
466             {
467                 Status = AcpiNsCheckObjectType (Info, &SubElements[j],
468                             Package->RetInfo2.ObjectType[j], j);
469                 if (ACPI_FAILURE (Status))
470                 {
471                     return (Status);
472                 }
473             }
474             break;
476         case ACPI_PTYPE2_MIN:
478             /* Each sub-package has a variable but minimum length */
480             ExpectedCount = Package->RetInfo.Count1;
481             if (SubPackage->Package.Count < ExpectedCount)
482             {
483                 goto PackageTooSmall;
484             }
486             /* Check the type of each sub-package element */
488             Status = AcpiNsCheckPackageElements (Info, SubElements,
489                         Package->RetInfo.ObjectType1,
490                         SubPackage->Package.Count, 0, 0, 0);
491             if (ACPI_FAILURE (Status))
492             {
493                 return (Status);
494             }
495             break;
497         case ACPI_PTYPE2_COUNT:
498             /*
499              * First element is the (Integer) count of elements, including
500              * the count field (the ACPI name is NumElements)
501              */
502             Status = AcpiNsCheckObjectType (Info, SubElements,
503                         ACPI_RTYPE_INTEGER, 0);
504             if (ACPI_FAILURE (Status))
505             {
506                 return (Status);
507             }
509             /*
510              * Make sure package is large enough for the Count and is
511              * is as large as the minimum size
512              */
513             ExpectedCount = (UINT32) (*SubElements)->Integer.Value;
514             if (SubPackage->Package.Count < ExpectedCount)
515             {
516                 goto PackageTooSmall;
517             }
518             if (SubPackage->Package.Count < Package->RetInfo.Count1)
519             {
520                 ExpectedCount = Package->RetInfo.Count1;
521                 goto PackageTooSmall;
522             }
523             if (ExpectedCount == 0)
```

```
524             {
525                 /*
526                  * Either the NumEntries element was originally zero or it was
527                  * a NULL element and repaired to an Integer of value zero.
528                  * In either case, repair it by setting NumEntries to be the
529                  * actual size of the subpackage.
530                  */
531                 ExpectedCount = SubPackage->Package.Count;
532                 (*SubElements)->Integer.Value = ExpectedCount;
533             }
534
535             /* Check the type of each sub-package element */
536
537             Status = AcpiNsCheckPackageElements (Info, (SubElements + 1),
538                         Package->RetInfo.ObjectType1,
539                         (ExpectedCount - 1), 0, 0, 1);
540             if (ACPI_FAILURE (Status))
541             {
542                 return (Status);
543             }
544             break;
545
546         default: /* Should not get here, type was validated by caller */
547
548             return (AE_AML_INTERNAL);
549         }
550
551         Elements++;
552     }
553
554     return (AE_OK);
555
556
557 PackageTooSmall:
558
559     /* The sub-package count was smaller than required */
560
561     ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
562         "Return Sub-Package[%u] is too small - found %u elements, expected %u",
563         i, SubPackage->Package.Count, ExpectedCount));
564
565     return (AE_AML_OPERAND_VALUE);
566 }
567
568
569 /******************************************************************************
570  *
571  * FUNCTION:    AcpiNsCheckPackageElements
572  *
573  * PARAMETERS:  Info            - Method execution information block
574  *              Elements        - Pointer to the package elements array
575  *              Type1           - Object type for first group
576  *              Count1          - Count for first group
577  *              Type2           - Object type for second group
578  *              Count2          - Count for second group
579  *              StartIndex      - Start of the first group of elements
580  *
581  * RETURN:      Status
582  *
583  * DESCRIPTION: Check that all elements of a package are of the correct object
584  *              type. Supports up to two groups of different object types.
585  *
586  *****************************************************************************/
587
588 static ACPI_STATUS
589 AcpiNsCheckPackageElements (
```



```
590     ACPI_EVALUATE_INFO          *Info,
591     ACPI_OPERAND_OBJECT         **Elements,
592     UINT8                       Type1,
593     UINT32                      Count1,
594     UINT8                       Type2,
595     UINT32                      Count2,
596     UINT32                      StartIndex)
597 {
598     ACPI_OPERAND_OBJECT         **ThisElement = Elements;
599     ACPI_STATUS                 Status;
600     UINT32                      i;
601
602
603     /*
604      * Up to two groups of package elements are supported by the data
605      * structure. All elements in each group must be of the same type.
606      * The second group can have a count of zero.
607      */
608     for (i = 0; i < Count1; i++)
609     {
610         Status = AcpiNsCheckObjectType (Info, ThisElement,
611                     Type1, i + StartIndex);
612         if (ACPI_FAILURE (Status))
613         {
614             return (Status);
615         }
616         ThisElement++;
617     }
618
619     for (i = 0; i < Count2; i++)
620     {
621         Status = AcpiNsCheckObjectType (Info, ThisElement,
622                     Type2, (i + Count1 + StartIndex));
623         if (ACPI_FAILURE (Status))
624         {
625             return (Status);
626         }
627         ThisElement++;
628     }
629
630     return (AE_OK);
631 }
```

```
**********************************************************
   20212 Thu Dec 26 13:49:20 2013
new/usr/src/common/acpica/components/namespace/nsrepair.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsrepair - Repair for objects returned by predefined methods
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSREPAIR_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acinterp.h"
#include "acpredef.h"
#include "amlresrc.h"

#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsrepair")


/*******************************************************************************
 *
 * This module attempts to repair or convert objects returned by the
 * predefined methods to an object type that is expected, as per the ACPI
```



```
 * specification. The need for this code is dictated by the many machines that
 * return incorrect types for the standard predefined methods. Performing these
 * conversions here, in one place, eliminates the need for individual ACPI
 * device drivers to do the same. Note: Most of these conversions are different
 * than the internal object conversion routines used for implicit object
 * conversion.
 *
 * The following conversions can be performed as necessary:
 *
 * Integer -> String
 * Integer -> Buffer
 * String  -> Integer
 * String  -> Buffer
 * Buffer  -> Integer
 * Buffer  -> String
 * Buffer  -> Package of Integers
 * Package -> Package of one Package
 *
 * Additional conversions that are available:
 *  Convert a null return or zero return value to an EndTag descriptor
 *  Convert an ASCII string to a Unicode buffer
 *
 * An incorrect standalone object is wrapped with required outer package
 *
 * Additional possible repairs:
 * Required package elements that are NULL replaced by Integer/String/Buffer
 *
 ******************************************************************************/


/* Local prototypes */

static const ACPI_SIMPLE_REPAIR_INFO *
AcpiNsMatchSimpleRepair (
    ACPI_NAMESPACE_NODE     *Node,
    UINT32                  ReturnBtype,
    UINT32                  PackageIndex);


/*
 * Special but simple repairs for some names.
 *
 * 2nd argument: Unexpected types that can be repaired
 */
static const ACPI_SIMPLE_REPAIR_INFO    AcpiObjectRepairInfo[] =
{
    /* Resource descriptor conversions */

    { "_CRS", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
                ACPI_NOT_PACKAGE_ELEMENT,
                AcpiNsConvertToResource },
    { "_DMA", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
                ACPI_NOT_PACKAGE_ELEMENT,
                AcpiNsConvertToResource },
    { "_PRS", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
                ACPI_NOT_PACKAGE_ELEMENT,
                AcpiNsConvertToResource },

    /* Unicode conversions */

    { "_MLS", ACPI_RTYPE_STRING, 1,
                AcpiNsConvertToUnicode },
    { "_STR", ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER,
                ACPI_NOT_PACKAGE_ELEMENT,
                AcpiNsConvertToUnicode },
    { {0,0,0,0}, 0, 0, NULL } /* Table terminator */
```

```
};


/*******************************************************************************
 *
 * FUNCTION:    AcpiNsSimpleRepair
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ExpectedBtypes      - Object types expected
 *              PackageIndex        - Index of object within parent package (if
 *                                    applicable - ACPI_NOT_PACKAGE_ELEMENT
 *                                    otherwise)
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if repair was successful.
 *
 * DESCRIPTION: Attempt to repair/convert a return object of a type that was
 *              not expected.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiNsSimpleRepair (
    ACPI_EVALUATE_INFO      *Info,
    UINT32                  ExpectedBtypes,
    UINT32                  PackageIndex,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
    ACPI_OPERAND_OBJECT     *NewObject = NULL;
    ACPI_STATUS             Status;
    const ACPI_SIMPLE_REPAIR_INFO   *Predefined;


    ACPI_FUNCTION_NAME (NsSimpleRepair);


    /*
     * Special repairs for certain names that are in the repair table.
     * Check if this name is in the list of repairable names.
     */
    Predefined = AcpiNsMatchSimpleRepair (Info->Node,
        Info->ReturnBtype, PackageIndex);
    if (Predefined)
    {
        if (!ReturnObject)
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname,
                ACPI_WARN_ALWAYS, "Missing expected return value"));
        }

        Status = Predefined->ObjectConverter (ReturnObject, &NewObject);
        if (ACPI_FAILURE (Status))
        {
            /* A fatal error occurred during a conversion */

            ACPI_EXCEPTION ((AE_INFO, Status,
                "During return object analysis"));
            return (Status);
        }
        if (NewObject)
        {
            goto ObjectRepaired;
        }
    }
```



```
    /*
     * Do not perform simple object repair unless the return type is not
     * expected.
     */
    if (Info->ReturnBtype & ExpectedBtypes)
    {
        return (AE_OK);
    }

    /*
     * At this point, we know that the type of the returned object was not
     * one of the expected types for this predefined name. Attempt to
     * repair the object by converting it to one of the expected object
     * types for this predefined name.
     */

    /*
     * If there is no return value, check if we require a return value for
     * this predefined name. Either one return value is expected, or none,
     * for both methods and other objects.
     *
     * Exit now if there is no return object. Warning if one was expected.
     */
    if (!ReturnObject)
    {
        if (ExpectedBtypes && (!(ExpectedBtypes & ACPI_RTYPE_NONE)))
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname,
                ACPI_WARN_ALWAYS, "Missing expected return value"));

            return (AE_AML_NO_RETURN_VALUE);
        }
    }

    if (ExpectedBtypes & ACPI_RTYPE_INTEGER)
    {
        Status = AcpiNsConvertToInteger (ReturnObject, &NewObject);
        if (ACPI_SUCCESS (Status))
        {
            goto ObjectRepaired;
        }
    }
    if (ExpectedBtypes & ACPI_RTYPE_STRING)
    {
        Status = AcpiNsConvertToString (ReturnObject, &NewObject);
        if (ACPI_SUCCESS (Status))
        {
            goto ObjectRepaired;
        }
    }
    if (ExpectedBtypes & ACPI_RTYPE_BUFFER)
    {
        Status = AcpiNsConvertToBuffer (ReturnObject, &NewObject);
        if (ACPI_SUCCESS (Status))
        {
            goto ObjectRepaired;
        }
    }
    if (ExpectedBtypes & ACPI_RTYPE_PACKAGE)
    {
        /*
         * A package is expected. We will wrap the existing object with a
         * new package object. It is often the case that if a variable-length
         * package is required, but there is only a single object needed, the
         * BIOS will return that object instead of wrapping it with a Package
```

```
259          * object. Note: after the wrapping, the package will be validated
260          * for correct contents (expected object type or types).
261          */
262         Status = AcpiNsWrapWithPackage (Info, ReturnObject, &NewObject);
263         if (ACPI_SUCCESS (Status))
264         {
265             /*
266              * The original object just had its reference count
267              * incremented for being inserted into the new package.
268              */
269             *ReturnObjectPtr = NewObject;       /* New Package object */
270             Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
271             return (AE_OK);
272         }
273     }

275     /* We cannot repair this object */

277     return (AE_AML_OPERAND_TYPE);


280 ObjectRepaired:

282     /* Object was successfully repaired */

284     if (PackageIndex != ACPI_NOT_PACKAGE_ELEMENT)
285     {
286         /*
287          * The original object is a package element. We need to
288          * decrement the reference count of the original object,
289          * for removing it from the package.
290          *
291          * However, if the original object was just wrapped with a
292          * package object as part of the repair, we don't need to
293          * change the reference count.
294          */
295         if (!(Info->ReturnFlags & ACPI_OBJECT_WRAPPED))
296         {
297             NewObject->Common.ReferenceCount =
298                 ReturnObject->Common.ReferenceCount;

300             if (ReturnObject->Common.ReferenceCount > 1)
301             {
302                 ReturnObject->Common.ReferenceCount--;
303             }
304         }

306         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
307             "%s: Converted %s to expected %s at Package index %u\n",
308             Info->FullPathname, AcpiUtGetObjectTypeName (ReturnObject),
309             AcpiUtGetObjectTypeName (NewObject), PackageIndex));
310     }
311     else
312     {
313         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
314             "%s: Converted %s to expected %s\n",
315             Info->FullPathname, AcpiUtGetObjectTypeName (ReturnObject),
316             AcpiUtGetObjectTypeName (NewObject)));
317     }

319     /* Delete old object, install the new return object */

321     AcpiUtRemoveReference (ReturnObject);
322     *ReturnObjectPtr = NewObject;
323     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
324     return (AE_OK);
```



```
325 }


328 /******************************************************************************
329  *
330  * FUNCTION:    AcpiNsMatchSimpleRepair
331  *
332  * PARAMETERS:  Node                - Namespace node for the method/object
333  *              ReturnBtype         - Object type that was returned
334  *              PackageIndex        - Index of object within parent package (if
335  *                                    applicable - ACPI_NOT_PACKAGE_ELEMENT
336  *                                    otherwise)
337  *
338  * RETURN:      Pointer to entry in repair table. NULL indicates not found.
339  *
340  * DESCRIPTION: Check an object name against the repairable object list.
341  *
342  *****************************************************************************/

344 static const ACPI_SIMPLE_REPAIR_INFO *
345 AcpiNsMatchSimpleRepair (
346     ACPI_NAMESPACE_NODE     *Node,
347     UINT32                  ReturnBtype,
348     UINT32                  PackageIndex)
349 {
350     const ACPI_SIMPLE_REPAIR_INFO   *ThisName;


353     /* Search info table for a repairable predefined method/object name */

355     ThisName = AcpiObjectRepairInfo;
356     while (ThisName->ObjectConverter)
357     {
358         if (ACPI_COMPARE_NAME (Node->Name.Ascii, ThisName->Name))
359         {
360             /* Check if we can actually repair this name/type combination */

362             if ((ReturnBtype & ThisName->UnexpectedBtypes) &&
363                 (PackageIndex == ThisName->PackageIndex))
364             {
365                 return (ThisName);
366             }

368             return (NULL);
369         }
370         ThisName++;
371     }

373     return (NULL); /* Name was not found in the repair table */
374 }


377 /******************************************************************************
378  *
379  * FUNCTION:    AcpiNsRepairNullElement
380  *
381  * PARAMETERS:  Info                - Method execution information block
382  *              ExpectedBtypes      - Object types expected
383  *              PackageIndex        - Index of object within parent package (if
384  *                                    applicable - ACPI_NOT_PACKAGE_ELEMENT
385  *                                    otherwise)
386  *              ReturnObjectPtr     - Pointer to the object returned from the
387  *                                    evaluation of a method or object
388  *
389  * RETURN:      Status. AE_OK if repair was successful.
390  *
```

```
391  * DESCRIPTION: Attempt to repair a NULL element of a returned Package object.
392  *
393  *****************************************************************************/

395 ACPI_STATUS
396 AcpiNsRepairNullElement (
397     ACPI_EVALUATE_INFO      *Info,
398     UINT32                  ExpectedBtypes,
399     UINT32                  PackageIndex,
400     ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
401 {
402     ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
403     ACPI_OPERAND_OBJECT     *NewObject;


406     ACPI_FUNCTION_NAME (NsRepairNullElement);


409     /* No repair needed if return object is non-NULL */

411     if (ReturnObject)
412     {
413         return (AE_OK);
414     }

416     /*
417      * Attempt to repair a NULL element of a Package object. This applies to
418      * predefined names that return a fixed-length package and each element
419      * is required. It does not apply to variable-length packages where NULL
420      * elements are allowed, especially at the end of the package.
421      */
422     if (ExpectedBtypes & ACPI_RTYPE_INTEGER)
423     {
424         /* Need an Integer - create a zero-value integer */

426         NewObject = AcpiUtCreateIntegerObject ((UINT64) 0);
427     }
428     else if (ExpectedBtypes & ACPI_RTYPE_STRING)
429     {
430         /* Need a String - create a NULL string */

432         NewObject = AcpiUtCreateStringObject (0);
433     }
434     else if (ExpectedBtypes & ACPI_RTYPE_BUFFER)
435     {
436         /* Need a Buffer - create a zero-length buffer */

438         NewObject = AcpiUtCreateBufferObject (0);
439     }
440     else
441     {
442         /* Error for all other expected types */

444         return (AE_AML_OPERAND_TYPE);
445     }

447     if (!NewObject)
448     {
449         return (AE_NO_MEMORY);
450     }

452     /* Set the reference count according to the parent Package object */

454     NewObject->Common.ReferenceCount = Info->ParentPackage->Common.ReferenceCoun

456     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
```



```
457         "%s: Converted NULL package element to expected %s at index %u\n",
458         Info->FullPathname, AcpiUtGetObjectTypeName (NewObject), PackageIndex));

460     *ReturnObjectPtr = NewObject;
461     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
462     return (AE_OK);
463 }


466 /*****************************************************************************
467  *
468  * FUNCTION:    AcpiNsRemoveNullElements
469  *
470  * PARAMETERS:  Info                - Method execution information block
471  *              PackageType         - An AcpiReturnPackageTypes value
472  *              ObjDesc             - A Package object
473  *
474  * RETURN:      None.
475  *
476  * DESCRIPTION: Remove all NULL package elements from packages that contain
477  *              a variable number of sub-packages. For these types of
478  *              packages, NULL elements can be safely removed.
479  *
480  *****************************************************************************/

482 void
483 AcpiNsRemoveNullElements (
484     ACPI_EVALUATE_INFO      *Info,
485     UINT8                   PackageType,
486     ACPI_OPERAND_OBJECT     *ObjDesc)
487 {
488     ACPI_OPERAND_OBJECT     **Source;
489     ACPI_OPERAND_OBJECT     **Dest;
490     UINT32                  Count;
491     UINT32                  NewCount;
492     UINT32                  i;


495     ACPI_FUNCTION_NAME (NsRemoveNullElements);


498     /*
499      * We can safely remove all NULL elements from these package types:
500      * PTYPE1_VAR packages contain a variable number of simple data types.
501      * PTYPE2 packages contain a variable number of sub-packages.
502      */
503     switch (PackageType)
504     {
505     case ACPI_PTYPE1_VAR:
506     case ACPI_PTYPE2:
507     case ACPI_PTYPE2_COUNT:
508     case ACPI_PTYPE2_PKG_COUNT:
509     case ACPI_PTYPE2_FIXED:
510     case ACPI_PTYPE2_MIN:
511     case ACPI_PTYPE2_REV_FIXED:
512     case ACPI_PTYPE2_FIX_VAR:

514         break;

516     default:
517     case ACPI_PTYPE1_FIXED:
518     case ACPI_PTYPE1_OPTION:
519         return;
520     }

522     Count = ObjDesc->Package.Count;
```

```
523     NewCount = Count;
524
525     Source = ObjDesc->Package.Elements;
526     Dest = Source;
527
528     /* Examine all elements of the package object, remove nulls */
529
530     for (i = 0; i < Count; i++)
531     {
532         if (!*Source)
533         {
534             NewCount--;
535         }
536         else
537         {
538             *Dest = *Source;
539             Dest++;
540         }
541         Source++;
542     }
543
544     /* Update parent package if any null elements were removed */
545
546     if (NewCount < Count)
547     {
548         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
549             "%s: Found and removed %u NULL elements\n",
550             Info->FullPathname, (Count - NewCount)));
551
552         /* NULL terminate list and update the package count */
553
554         *Dest = NULL;
555         ObjDesc->Package.Count = NewCount;
556     }
557 }
558
559
560 /******************************************************************************
561  *
562  * FUNCTION:    AcpiNsWrapWithPackage
563  *
564  * PARAMETERS:  Info                - Method execution information block
565  *              OriginalObject      - Pointer to the object to repair.
566  *              ObjDescPtr          - The new package object is returned here
567  *
568  * RETURN:      Status, new object in *ObjDescPtr
569  *
570  * DESCRIPTION: Repair a common problem with objects that are defined to
571  *              return a variable-length Package of sub-objects. If there is
572  *              only one sub-object, some BIOS code mistakenly simply declares
573  *              the single object instead of a Package with one sub-object.
574  *              This function attempts to repair this error by wrapping a
575  *              Package object around the original object, creating the
576  *              correct and expected Package with one sub-object.
577  *
578  *              Names that can be repaired in this manner include:
579  *              _ALR, _CSD, _HPX, _MLS, _PLD, _PRT, _PSS, _TRT, _TSS,
580  *              _BCL, _DOD, _FIX, _Sx
581  *
582  *****************************************************************************/
583
584 ACPI_STATUS
585 AcpiNsWrapWithPackage (
586     ACPI_EVALUATE_INFO      *Info,
587     ACPI_OPERAND_OBJECT     *OriginalObject,
588     ACPI_OPERAND_OBJECT     **ObjDescPtr)
```



```
589 {
590     ACPI_OPERAND_OBJECT     *PkgObjDesc;
591
592
593     ACPI_FUNCTION_NAME (NsWrapWithPackage);
594
595
596     /*
597      * Create the new outer package and populate it. The new package will
598      * have a single element, the lone sub-object.
599      */
600     PkgObjDesc = AcpiUtCreatePackageObject (1);
601     if (!PkgObjDesc)
602     {
603         return (AE_NO_MEMORY);
604     }
605
606     PkgObjDesc->Package.Elements[0] = OriginalObject;
607
608     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
609         "%s: Wrapped %s with expected Package object\n",
610         Info->FullPathname, AcpiUtGetObjectTypeName (OriginalObject)));
611
612     /* Return the new object in the object pointer */
613
614     *ObjDescPtr = PkgObjDesc;
615     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED | ACPI_OBJECT_WRAPPED;
616     return (AE_OK);
617 }
```

```
**********************************************************
   33442 Thu Dec 26 13:49:21 2013
new/usr/src/common/acpica/components/namespace/nsrepair2.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsrepair2 - Repair for objects returned by specific
 *                          predefined methods
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __NSREPAIR2_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsrepair2")


/*
 * Information structure and handler for ACPI predefined names that can
 * be repaired on a per-name basis.
 */
typedef
ACPI_STATUS (*ACPI_REPAIR_FUNCTION) (
```



```
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

typedef struct acpi_repair_info
{
    char                    Name[ACPI_NAME_SIZE];
    ACPI_REPAIR_FUNCTION    RepairFunction;

} ACPI_REPAIR_INFO;


/* Local prototypes */

static const ACPI_REPAIR_INFO *
AcpiNsMatchComplexRepair (
    ACPI_NAMESPACE_NODE     *Node);

static ACPI_STATUS
AcpiNsRepair_ALR (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_CID (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_CST (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_FDE (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_HID (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_PRT (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_PSS (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsRepair_TSS (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

static ACPI_STATUS
AcpiNsCheckSortedList (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     *ReturnObject,
    UINT32                  StartIndex,
    UINT32                  ExpectedCount,
    UINT32                  SortIndex,
    UINT8                   SortDirection,
    char                    *SortKeyName);
```

```
/* Values for SortDirection above */

#define ACPI_SORT_ASCENDING     0
#define ACPI_SORT_DESCENDING    1

static void
AcpiNsRemoveElement (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  Index);

static void
AcpiNsSortList (
    ACPI_OPERAND_OBJECT     **Elements,
    UINT32                  Count,
    UINT32                  Index,
    UINT8                   SortDirection);


/*
 * This table contains the names of the predefined methods for which we can
 * perform more complex repairs.
 *
 * As necessary:
 *
 * _ALR: Sort the list ascending by AmbientIlluminance
 * _CID: Strings: uppercase all, remove any leading asterisk
 * _CST: Sort the list ascending by C state type
 * _FDE: Convert Buffer of BYTEs to a Buffer of DWORDs
 * _GTM: Convert Buffer of BYTEs to a Buffer of DWORDs
 * _HID: Strings: uppercase all, remove any leading asterisk
 * _PRT: Fix reversed SourceName and SourceIndex
 * _PSS: Sort the list descending by Power
 * _TSS: Sort the list descending by Power
 *
 * Names that must be packages, but cannot be sorted:
 *
 * _BCL: Values are tied to the Package index where they appear, and cannot
 * be moved or sorted. These index values are used for _BQC and _BCM.
 * However, we can fix the case where a buffer is returned, by converting
 * it to a Package of integers.
 */
static const ACPI_REPAIR_INFO       AcpiNsRepairableNames[] =
{
    {"_ALR", AcpiNsRepair_ALR},
    {"_CID", AcpiNsRepair_CID},
    {"_CST", AcpiNsRepair_CST},
    {"_FDE", AcpiNsRepair_FDE},
    {"_GTM", AcpiNsRepair_FDE},     /* _GTM has same repair as _FDE */
    {"_HID", AcpiNsRepair_HID},
    {"_PRT", AcpiNsRepair_PRT},
    {"_PSS", AcpiNsRepair_PSS},
    {"_TSS", AcpiNsRepair_TSS},
    {{0,0,0,0}, NULL}               /* Table terminator */
};


#define ACPI_FDE_FIELD_COUNT        5
#define ACPI_FDE_BYTE_BUFFER_SIZE   5
#define ACPI_FDE_DWORD_BUFFER_SIZE  (ACPI_FDE_FIELD_COUNT * sizeof (UINT32))


/******************************************************************************
 *
 * FUNCTION:    AcpiNsComplexRepairs
 *
```



```
 * PARAMETERS:  Info                - Method execution information block
 *              Node                - Namespace node for the method/object
 *              ValidateStatus      - Original status of earlier validation
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if repair was successful. If name is not
 *              matched, ValidateStatus is returned.
 *
 * DESCRIPTION: Attempt to repair/convert a return object of a type that was
 *              not expected.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiNsComplexRepairs (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_STATUS             ValidateStatus,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    const ACPI_REPAIR_INFO  *Predefined;
    ACPI_STATUS             Status;


    /* Check if this name is in the list of repairable names */

    Predefined = AcpiNsMatchComplexRepair (Node);
    if (!Predefined)
    {
        return (ValidateStatus);
    }

    Status = Predefined->RepairFunction (Info, ReturnObjectPtr);
    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiNsMatchComplexRepair
 *
 * PARAMETERS:  Node                - Namespace node for the method/object
 *
 * RETURN:      Pointer to entry in repair table. NULL indicates not found.
 *
 * DESCRIPTION: Check an object name against the repairable object list.
 *
 *****************************************************************************/

static const ACPI_REPAIR_INFO *
AcpiNsMatchComplexRepair (
    ACPI_NAMESPACE_NODE     *Node)
{
    const ACPI_REPAIR_INFO  *ThisName;


    /* Search info table for a repairable predefined method/object name */

    ThisName = AcpiNsRepairableNames;
    while (ThisName->RepairFunction)
    {
        if (ACPI_COMPARE_NAME (Node->Name.Ascii, ThisName->Name))
        {
            return (ThisName);
        }
```

```
259         ThisName++;
260     }

262     return (NULL); /* Not found */
263 }


266 /*****************************************************************************
267  *
268  * FUNCTION:    AcpiNsRepair_ALR
269  *
270  * PARAMETERS:  Info                - Method execution information block
271  *              ReturnObjectPtr     - Pointer to the object returned from the
272  *                                    evaluation of a method or object
273  *
274  * RETURN:      Status. AE_OK if object is OK or was repaired successfully
275  *
276  * DESCRIPTION: Repair for the _ALR object. If necessary, sort the object list
277  *              ascending by the ambient illuminance values.
278  *
279  ****************************************************************************/

281 static ACPI_STATUS
282 AcpiNsRepair_ALR (
283     ACPI_EVALUATE_INFO      *Info,
284     ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
285 {
286     ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
287     ACPI_STATUS             Status;


290     Status = AcpiNsCheckSortedList (Info, ReturnObject, 0, 2, 1,
291                 ACPI_SORT_ASCENDING, "AmbientIlluminance");

293     return (Status);
294 }


297 /*****************************************************************************
298  *
299  * FUNCTION:    AcpiNsRepair_FDE
300  *
301  * PARAMETERS:  Info                - Method execution information block
302  *              ReturnObjectPtr     - Pointer to the object returned from the
303  *                                    evaluation of a method or object
304  *
305  * RETURN:      Status. AE_OK if object is OK or was repaired successfully
306  *
307  * DESCRIPTION: Repair for the _FDE and _GTM objects. The expected return
308  *              value is a Buffer of 5 DWORDs. This function repairs a common
309  *              problem where the return value is a Buffer of BYTEs, not
310  *              DWORDs.
311  *
312  ****************************************************************************/

314 static ACPI_STATUS
315 AcpiNsRepair_FDE (
316     ACPI_EVALUATE_INFO      *Info,
317     ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
318 {
319     ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
320     ACPI_OPERAND_OBJECT     *BufferObject;
321     UINT8                   *ByteBuffer;
322     UINT32                  *DwordBuffer;
323     UINT32                  i;
```



```
326     ACPI_FUNCTION_NAME (NsRepair_FDE);


329     switch (ReturnObject->Common.Type)
330     {
331     case ACPI_TYPE_BUFFER:

333         /* This is the expected type. Length should be (at least) 5 DWORDs */

335         if (ReturnObject->Buffer.Length >= ACPI_FDE_DWORD_BUFFER_SIZE)
336         {
337             return (AE_OK);
338         }

340         /* We can only repair if we have exactly 5 BYTEs */

342         if (ReturnObject->Buffer.Length != ACPI_FDE_BYTE_BUFFER_SIZE)
343         {
344             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
345                 "Incorrect return buffer length %u, expected %u",
346                 ReturnObject->Buffer.Length, ACPI_FDE_DWORD_BUFFER_SIZE));

348             return (AE_AML_OPERAND_TYPE);
349         }

351         /* Create the new (larger) buffer object */

353         BufferObject = AcpiUtCreateBufferObject (ACPI_FDE_DWORD_BUFFER_SIZE);
354         if (!BufferObject)
355         {
356             return (AE_NO_MEMORY);
357         }

359         /* Expand each byte to a DWORD */

361         ByteBuffer = ReturnObject->Buffer.Pointer;
362         DwordBuffer = ACPI_CAST_PTR (UINT32, BufferObject->Buffer.Pointer);

364         for (i = 0; i < ACPI_FDE_FIELD_COUNT; i++)
365         {
366             *DwordBuffer = (UINT32) *ByteBuffer;
367             DwordBuffer++;
368             ByteBuffer++;
369         }

371         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
372             "%s Expanded Byte Buffer to expected DWord Buffer\n",
373             Info->FullPathname));
374         break;

376     default:

378         return (AE_AML_OPERAND_TYPE);
379     }

381     /* Delete the original return object, return the new buffer object */

383     AcpiUtRemoveReference (ReturnObject);
384     *ReturnObjectPtr = BufferObject;

386     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
387     return (AE_OK);
388 }
```

```
/******************************************************************************
 *
 * FUNCTION:    AcpiNsRepair_CID
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
 *
 * DESCRIPTION: Repair for the _CID object. If a string, ensure that all
 *              letters are uppercase and that there is no leading asterisk.
 *              If a Package, ensure same for all string elements.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiNsRepair_CID (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
    ACPI_OPERAND_OBJECT     **ElementPtr;
    ACPI_OPERAND_OBJECT     *OriginalElement;
    UINT16                  OriginalRefCount;
    UINT32                  i;


    /* Check for _CID as a simple string */

    if (ReturnObject->Common.Type == ACPI_TYPE_STRING)
    {
        Status = AcpiNsRepair_HID (Info, ReturnObjectPtr);
        return (Status);
    }

    /* Exit if not a Package */

    if (ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
    {
        return (AE_OK);
    }

    /* Examine each element of the _CID package */

    ElementPtr = ReturnObject->Package.Elements;
    for (i = 0; i < ReturnObject->Package.Count; i++)
    {
        OriginalElement = *ElementPtr;
        OriginalRefCount = OriginalElement->Common.ReferenceCount;

        Status = AcpiNsRepair_HID (Info, ElementPtr);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }

        /* Take care with reference counts */

        if (OriginalElement != *ElementPtr)
        {
            /* Element was replaced */

            (*ElementPtr)->Common.ReferenceCount =
                OriginalRefCount;
```



```
            AcpiUtRemoveReference (OriginalElement);
        }

        ElementPtr++;
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiNsRepair_CST
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
 *
 * DESCRIPTION: Repair for the _CST object:
 *              1. Sort the list ascending by C state type
 *              2. Ensure type cannot be zero
 *              3. A sub-package count of zero means _CST is meaningless
 *              4. Count must match the number of C state sub-packages
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiNsRepair_CST (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
    ACPI_OPERAND_OBJECT     **OuterElements;
    UINT32                  OuterElementCount;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_STATUS             Status;
    BOOLEAN                 Removing;
    UINT32                  i;


    ACPI_FUNCTION_NAME (NsRepair_CST);


    /*
     * Check if the C-state type values are proportional.
     */
    OuterElementCount = ReturnObject->Package.Count - 1;
    i = 0;
    while (i < OuterElementCount)
    {
        OuterElements = &ReturnObject->Package.Elements[i + 1];
        Removing = FALSE;

        if ((*OuterElements)->Package.Count == 0)
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
                "SubPackage[%u] - removing entry due to zero count", i));
            Removing = TRUE;
            goto RemoveElement;
        }

        ObjDesc = (*OuterElements)->Package.Elements[1]; /* Index1 = Type */
        if ((UINT32) ObjDesc->Integer.Value == 0)
```

```
523         {
524             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
525                 "SubPackage[%u] - removing entry due to invalid Type(0)", i));
526             Removing = TRUE;
527         }
528
529 RemoveElement:
530         if (Removing)
531         {
532             AcpiNsRemoveElement (ReturnObject, i + 1);
533             OuterElementCount--;
534         }
535         else
536         {
537             i++;
538         }
539     }
540
541     /* Update top-level package count, Type "Integer" checked elsewhere */
542
543     ObjDesc = ReturnObject->Package.Elements[0];
544     ObjDesc->Integer.Value = OuterElementCount;
545
546     /*
547      * Entries (subpackages) in the _CST Package must be sorted by the
548      * C-state type, in ascending order.
549      */
550     Status = AcpiNsCheckSortedList (Info, ReturnObject, 1, 4, 1,
551                 ACPI_SORT_ASCENDING, "C-State Type");
552     if (ACPI_FAILURE (Status))
553     {
554         return (Status);
555     }
556
557     return (AE_OK);
558 }
559
560
561 /*******************************************************************************
562  *
563  * FUNCTION:    AcpiNsRepair_HID
564  *
565  * PARAMETERS:  Info                - Method execution information block
566  *              ReturnObjectPtr     - Pointer to the object returned from the
567  *                                    evaluation of a method or object
568  *
569  * RETURN:      Status. AE_OK if object is OK or was repaired successfully
570  *
571  * DESCRIPTION: Repair for the _HID object. If a string, ensure that all
572  *              letters are uppercase and that there is no leading asterisk.
573  *
574  ******************************************************************************/
575
576 static ACPI_STATUS
577 AcpiNsRepair_HID (
578     ACPI_EVALUATE_INFO      *Info,
579     ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
580 {
581     ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
582     ACPI_OPERAND_OBJECT     *NewString;
583     char                    *Source;
584     char                    *Dest;
585
586
587     ACPI_FUNCTION_NAME (NsRepair_HID);
```



```
588
589
590     /* We only care about string _HID objects (not integers) */
591
592     if (ReturnObject->Common.Type != ACPI_TYPE_STRING)
593     {
594         return (AE_OK);
595     }
596
597     if (ReturnObject->String.Length == 0)
598     {
599         ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
600             "Invalid zero-length _HID or _CID string"));
601
602         /* Return AE_OK anyway, let driver handle it */
603
604         Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
605         return (AE_OK);
606     }
607
608     /* It is simplest to always create a new string object */
609
610     NewString = AcpiUtCreateStringObject (ReturnObject->String.Length);
611     if (!NewString)
612     {
613         return (AE_NO_MEMORY);
614     }
615
616     /*
617      * Remove a leading asterisk if present. For some unknown reason, there
618      * are many machines in the field that contains IDs like this.
619      *
620      * Examples: "*PNP0C03", "*ACPI0003"
621      */
622     Source = ReturnObject->String.Pointer;
623     if (*Source == '*')
624     {
625         Source++;
626         NewString->String.Length--;
627
628         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
629             "%s: Removed invalid leading asterisk\n", Info->FullPathname));
630     }
631
632     /*
633      * Copy and uppercase the string. From the ACPI 5.0 specification:
634      *
635      * A valid PNP ID must be of the form "AAA####" where A is an uppercase
636      * letter and # is a hex digit. A valid ACPI ID must be of the form
637      * "NNNN####" where N is an uppercase letter or decimal digit, and
638      * # is a hex digit.
639      */
640     for (Dest = NewString->String.Pointer; *Source; Dest++, Source++)
641     {
642         *Dest = (char) ACPI_TOUPPER (*Source);
643     }
644
645     AcpiUtRemoveReference (ReturnObject);
646     *ReturnObjectPtr = NewString;
647     return (AE_OK);
648 }
649
650
651 /*******************************************************************************
652  *
653  * FUNCTION:    AcpiNsRepair_PRT
654  *
```

```
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
 *
 * DESCRIPTION: Repair for the _PRT object. If necessary, fix reversed
 *              SourceName and SourceIndex field, a common BIOS bug.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiNsRepair_PRT (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT     *PackageObject = *ReturnObjectPtr;
    ACPI_OPERAND_OBJECT     **TopObjectList;
    ACPI_OPERAND_OBJECT     **SubObjectList;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    UINT32                  ElementCount;
    UINT32                  Index;


    /* Each element in the _PRT package is a subpackage */

    TopObjectList = PackageObject->Package.Elements;
    ElementCount = PackageObject->Package.Count;

    for (Index = 0; Index < ElementCount; Index++)
    {
        SubObjectList = (*TopObjectList)->Package.Elements;

        /*
         * If the BIOS has erroneously reversed the _PRT SourceName (index 2)
         * and the SourceIndex (index 3), fix it. _PRT is important enough to
         * workaround this BIOS error. This also provides compatibility with
         * other ACPI implementations.
         */
        ObjDesc = SubObjectList[3];
        if (!ObjDesc || (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
        {
            SubObjectList[3] = SubObjectList[2];
            SubObjectList[2] = ObjDesc;
            Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;

            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
                "PRT[%X]: Fixed reversed SourceName and SourceIndex",
                Index));
        }

        /* Point to the next ACPI_OPERAND_OBJECT in the top level package */

        TopObjectList++;
    }

    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiNsRepair_PSS
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
```



```
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
 *
 * DESCRIPTION: Repair for the _PSS object. If necessary, sort the object list
 *              by the CPU frequencies. Check that the power dissipation values
 *              are all proportional to CPU frequency (i.e., sorting by
 *              frequency should be the same as sorting by power.)
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiNsRepair_PSS (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
    ACPI_OPERAND_OBJECT     **OuterElements;
    UINT32                  OuterElementCount;
    ACPI_OPERAND_OBJECT     **Elements;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    UINT32                  PreviousValue;
    ACPI_STATUS             Status;
    UINT32                  i;


    /*
     * Entries (sub-packages) in the _PSS Package must be sorted by power
     * dissipation, in descending order. If it appears that the list is
     * incorrectly sorted, sort it. We sort by CpuFrequency, since this
     * should be proportional to the power.
     */
    Status =AcpiNsCheckSortedList (Info, ReturnObject, 0, 6, 0,
                ACPI_SORT_DESCENDING, "CpuFrequency");
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /*
     * We now know the list is correctly sorted by CPU frequency. Check if
     * the power dissipation values are proportional.
     */
    PreviousValue = ACPI_UINT32_MAX;
    OuterElements = ReturnObject->Package.Elements;
    OuterElementCount = ReturnObject->Package.Count;

    for (i = 0; i < OuterElementCount; i++)
    {
        Elements = (*OuterElements)->Package.Elements;
        ObjDesc = Elements[1]; /* Index1 = PowerDissipation */

        if ((UINT32) ObjDesc->Integer.Value > PreviousValue)
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
                "SubPackage[%u,%u] - suspicious power dissipation values",
                i-1, i));
        }

        PreviousValue = (UINT32) ObjDesc->Integer.Value;
        OuterElements++;
    }

    return (AE_OK);
}
```

```
/*****************************************************************************
 *
 * FUNCTION:    AcpiNsRepair_TSS
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObjectPtr     - Pointer to the object returned from the
 *                                    evaluation of a method or object
 *
 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
 *
 * DESCRIPTION: Repair for the _TSS object. If necessary, sort the object list
 *              descending by the power dissipation values.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiNsRepair_TSS (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
{
    ACPI_OPERAND_OBJECT     *ReturnObject = *ReturnObjectPtr;
    ACPI_STATUS             Status;
    ACPI_NAMESPACE_NODE     *Node;


    /*
     * We can only sort the _TSS return package if there is no _PSS in the
     * same scope. This is because if _PSS is present, the ACPI specification
     * dictates that the _TSS Power Dissipation field is to be ignored, and
     * therefore some BIOSs leave garbage values in the _TSS Power field(s).
     * In this case, it is best to just return the _TSS package as-is.
     * (May, 2011)
     */
    Status = AcpiNsGetNode (Info->Node, "^_PSS",
        ACPI_NS_NO_UPSEARCH, &Node);
    if (ACPI_SUCCESS (Status))
    {
        return (AE_OK);
    }

    Status = AcpiNsCheckSortedList (Info, ReturnObject, 0, 5, 1,
                ACPI_SORT_DESCENDING, "PowerDissipation");

    return (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiNsCheckSortedList
 *
 * PARAMETERS:  Info                - Method execution information block
 *              ReturnObject        - Pointer to the top-level returned object
 *              StartIndex          - Index of the first sub-package
 *              ExpectedCount       - Minimum length of each sub-package
 *              SortIndex           - Sub-package entry to sort on
 *              SortDirection       - Ascending or descending
 *              SortKeyName         - Name of the SortIndex field
 *
 * RETURN:      Status. AE_OK if the list is valid and is sorted correctly or
 *              has been repaired by sorting the list.
 *
 * DESCRIPTION: Check if the package list is valid and sorted correctly by the
 *              SortIndex. If not, then sort the list.
 *
```



```
 *****************************************************************************/

static ACPI_STATUS
AcpiNsCheckSortedList (
    ACPI_EVALUATE_INFO      *Info,
    ACPI_OPERAND_OBJECT     *ReturnObject,
    UINT32                  StartIndex,
    UINT32                  ExpectedCount,
    UINT32                  SortIndex,
    UINT8                   SortDirection,
    char                    *SortKeyName)
{
    UINT32                  OuterElementCount;
    ACPI_OPERAND_OBJECT     **OuterElements;
    ACPI_OPERAND_OBJECT     **Elements;
    ACPI_OPERAND_OBJECT     *ObjDesc;
    UINT32                  i;
    UINT32                  PreviousValue;


    ACPI_FUNCTION_NAME (NsCheckSortedList);


    /* The top-level object must be a package */

    if (ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
    {
        return (AE_AML_OPERAND_TYPE);
    }

    /*
     * NOTE: assumes list of sub-packages contains no NULL elements.
     * Any NULL elements should have been removed by earlier call
     * to AcpiNsRemoveNullElements.
     */
    OuterElementCount = ReturnObject->Package.Count;
    if (!OuterElementCount || StartIndex >= OuterElementCount)
    {
        return (AE_AML_PACKAGE_LIMIT);
    }

    OuterElements = &ReturnObject->Package.Elements[StartIndex];
    OuterElementCount -= StartIndex;

    PreviousValue = 0;
    if (SortDirection == ACPI_SORT_DESCENDING)
    {
        PreviousValue = ACPI_UINT32_MAX;
    }

    /* Examine each subpackage */

    for (i = 0; i < OuterElementCount; i++)
    {
        /* Each element of the top-level package must also be a package */

        if ((*OuterElements)->Common.Type != ACPI_TYPE_PACKAGE)
        {
            return (AE_AML_OPERAND_TYPE);
        }

        /* Each sub-package must have the minimum length */

        if ((*OuterElements)->Package.Count < ExpectedCount)
        {
            return (AE_AML_PACKAGE_LIMIT);
```

```
        }

        Elements = (*OuterElements)->Package.Elements;
        ObjDesc = Elements[SortIndex];

        if (ObjDesc->Common.Type != ACPI_TYPE_INTEGER)
        {
            return (AE_AML_OPERAND_TYPE);
        }

        /*
         * The list must be sorted in the specified order. If we detect a
         * discrepancy, sort the entire list.
         */
        if (((SortDirection == ACPI_SORT_ASCENDING) &&
                (ObjDesc->Integer.Value < PreviousValue)) ||
            ((SortDirection == ACPI_SORT_DESCENDING) &&
                (ObjDesc->Integer.Value > PreviousValue)))
        {
            AcpiNsSortList (&ReturnObject->Package.Elements[StartIndex],
                OuterElementCount, SortIndex, SortDirection);

            Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;

            ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
                "%s: Repaired unsorted list - now sorted by %s\n",
                Info->FullPathname, SortKeyName));
            return (AE_OK);
        }

        PreviousValue = (UINT32) ObjDesc->Integer.Value;
        OuterElements++;
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiNsSortList
 *
 * PARAMETERS:  Elements            - Package object element list
 *              Count               - Element count for above
 *              Index               - Sort by which package element
 *              SortDirection       - Ascending or Descending sort
 *
 * RETURN:      None
 *
 * DESCRIPTION: Sort the objects that are in a package element list.
 *
 * NOTE: Assumes that all NULL elements have been removed from the package,
 *       and that all elements have been verified to be of type Integer.
 *
 *****************************************************************************/

static void
AcpiNsSortList (
    ACPI_OPERAND_OBJECT     **Elements,
    UINT32                  Count,
    UINT32                  Index,
    UINT8                   SortDirection)
{
    ACPI_OPERAND_OBJECT     *ObjDesc1;
    ACPI_OPERAND_OBJECT     *ObjDesc2;
    ACPI_OPERAND_OBJECT     *TempObj;
```



```
    UINT32                  i;
    UINT32                  j;


    /* Simple bubble sort */

    for (i = 1; i < Count; i++)
    {
        for (j = (Count - 1); j >= i; j--)
        {
            ObjDesc1 = Elements[j-1]->Package.Elements[Index];
            ObjDesc2 = Elements[j]->Package.Elements[Index];

            if (((SortDirection == ACPI_SORT_ASCENDING) &&
                    (ObjDesc1->Integer.Value > ObjDesc2->Integer.Value)) ||

                ((SortDirection == ACPI_SORT_DESCENDING) &&
                    (ObjDesc1->Integer.Value < ObjDesc2->Integer.Value)))
            {
                TempObj = Elements[j-1];
                Elements[j-1] = Elements[j];
                Elements[j] = TempObj;
            }
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiNsRemoveElement
 *
 * PARAMETERS:  ObjDesc             - Package object element list
 *              Index               - Index of element to remove
 *
 * RETURN:      None
 *
 * DESCRIPTION: Remove the requested element of a package and delete it.
 *
 *****************************************************************************/

static void
AcpiNsRemoveElement (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT32                  Index)
{
    ACPI_OPERAND_OBJECT     **Source;
    ACPI_OPERAND_OBJECT     **Dest;
    UINT32                  Count;
    UINT32                  NewCount;
    UINT32                  i;


    ACPI_FUNCTION_NAME (NsRemoveElement);


    Count = ObjDesc->Package.Count;
    NewCount = Count - 1;

    Source = ObjDesc->Package.Elements;
    Dest = Source;

    /* Examine all elements of the package object, remove matched index */

    for (i = 0; i < Count; i++)
    {
```

```
1051         if (i == Index)
1052         {
1053             AcpiUtRemoveReference (*Source); /* Remove one ref for being in pkg
1054             AcpiUtRemoveReference (*Source);
1055         }
1056         else
1057         {
1058             *Dest = *Source;
1059             Dest++;
1060         }
1061         Source++;
1062     }

1064     /* NULL terminate list and update the package count */

1066     *Dest = NULL;
1067     ObjDesc->Package.Count = NewCount;
1068 }
```

```
**********************************************************
   14383 Thu Dec 26 13:49:21 2013
new/usr/src/common/acpica/components/namespace/nssearch.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: nssearch - Namespace search
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __NSSEARCH_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acnamesp.h"

  50 #ifdef ACPI_ASL_COMPILER
  51 #include "amlcode.h"
  52 #endif

  54 #define _COMPONENT          ACPI_NAMESPACE
  55         ACPI_MODULE_NAME    ("nssearch")

  57 /* Local prototypes */

  59 static ACPI_STATUS
```



```
  60 AcpiNsSearchParentTree (
  61     UINT32                  TargetName,
  62     ACPI_NAMESPACE_NODE     *Node,
  63     ACPI_OBJECT_TYPE        Type,
  64     ACPI_NAMESPACE_NODE     **ReturnNode);


  67 /******************************************************************************
  68  *
  69  * FUNCTION:    AcpiNsSearchOneScope
  70  *
  71  * PARAMETERS:  TargetName      - Ascii ACPI name to search for
  72  *              ParentNode      - Starting node where search will begin
  73  *              Type            - Object type to match
  74  *              ReturnNode      - Where the matched Named obj is returned
  75  *
  76  * RETURN:      Status
  77  *
  78  * DESCRIPTION: Search a single level of the namespace. Performs a
  79  *              simple search of the specified level, and does not add
  80  *              entries or search parents.
  81  *
  82  *
  83  *      Named object lists are built (and subsequently dumped) in the
  84  *      order in which the names are encountered during the namespace load;
  85  *
  86  *      All namespace searching is linear in this implementation, but
  87  *      could be easily modified to support any improved search
  88  *      algorithm. However, the linear search was chosen for simplicity
  89  *      and because the trees are small and the other interpreter
  90  *      execution overhead is relatively high.
  91  *
  92  *      Note: CPU execution analysis has shown that the AML interpreter spends
  93  *      a very small percentage of its time searching the namespace. Therefore,
  94  *      the linear search seems to be sufficient, as there would seem to be
  95  *      little value in improving the search.
  96  *
  97  *****************************************************************************/

  99 ACPI_STATUS
 100 AcpiNsSearchOneScope (
 101     UINT32                  TargetName,
 102     ACPI_NAMESPACE_NODE     *ParentNode,
 103     ACPI_OBJECT_TYPE        Type,
 104     ACPI_NAMESPACE_NODE     **ReturnNode)
 105 {
 106     ACPI_NAMESPACE_NODE     *Node;


 109     ACPI_FUNCTION_TRACE (NsSearchOneScope);


 112 #ifdef ACPI_DEBUG_OUTPUT
 113     if (ACPI_LV_NAMES & AcpiDbgLevel)
 114     {
 115         char                *ScopeName;

 117         ScopeName = AcpiNsGetExternalPathname (ParentNode);
 118         if (ScopeName)
 119         {
 120             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
 121                 "Searching %s (%p) For [%4.4s] (%s)\n",
 122                 ScopeName, ParentNode, ACPI_CAST_PTR (char, &TargetName),
 123                 AcpiUtGetTypeName (Type)));

 125             ACPI_FREE (ScopeName);
```

```
126         }
127     }
128 #endif

130     /*
131      * Search for name at this namespace level, which is to say that we
132      * must search for the name among the children of this object
133      */
134     Node = ParentNode->Child;
135     while (Node)
136     {
137         /* Check for match against the name */

139         if (Node->Name.Integer == TargetName)
140         {
141             /* Resolve a control method alias if any */

143             if (AcpiNsGetType (Node) == ACPI_TYPE_LOCAL_METHOD_ALIAS)
144             {
145                 Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Node->Object);
146             }

148             /* Found matching entry */

150             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
151                 "Name [%4.4s] (%s) %p found in scope [%4.4s] %p\n",
152                 ACPI_CAST_PTR (char, &TargetName),
153                 AcpiUtGetTypeName (Node->Type),
154                 Node, AcpiUtGetNodeName (ParentNode), ParentNode));

156             *ReturnNode = Node;
157             return_ACPI_STATUS (AE_OK);
158         }

160         /* Didn't match name, move on to the next peer object */

162         Node = Node->Peer;
163     }

165     /* Searched entire namespace level, not found */

167     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
168         "Name [%4.4s] (%s) not found in search in scope [%4.4s] "
169         "%p first child %p\n",
170         ACPI_CAST_PTR (char, &TargetName), AcpiUtGetTypeName (Type),
171         AcpiUtGetNodeName (ParentNode), ParentNode, ParentNode->Child));

173     return_ACPI_STATUS (AE_NOT_FOUND);
174 }
```

unchanged_portion_omitted_

```
**********************************************************
   22821 Thu Dec 26 13:49:21 2013
new/usr/src/common/acpica/components/namespace/nsutils.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: nsutils - Utilities for accessing ACPI namespace, accessing
   4  *                        parents and siblings and Scope manipulation
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  45 #define __NSUTILS_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acnamesp.h"
  50 #include "amlcode.h"

  52 #define _COMPONENT          ACPI_NAMESPACE
  53         ACPI_MODULE_NAME    ("nsutils")

  55 /* Local prototypes */

  57 static BOOLEAN
  58 AcpiNsValidPathSeparator (
```



```
  59     char                    Sep);

  57 #ifdef ACPI_OBSOLETE_FUNCTIONS
  58 ACPI_NAME
  59 AcpiNsFindParentName (
  60     ACPI_NAMESPACE_NODE     *NodeToSearch);
  61 #endif


  64 /******************************************************************************
  65  *
  66  * FUNCTION:    AcpiNsPrintNodePathname
  67  *
  68  * PARAMETERS:  Node            - Object
  69  *              Message         - Prefix message
  70  *
  71  * DESCRIPTION: Print an object's full namespace pathname
  72  *              Manages allocation/freeing of a pathname buffer
  73  *
  74  *****************************************************************************/

  76 void
  77 AcpiNsPrintNodePathname (
  78     ACPI_NAMESPACE_NODE     *Node,
  79     const char              *Message)
  80 {
  81     ACPI_BUFFER             Buffer;
  82     ACPI_STATUS             Status;


  85     if (!Node)
  86     {
  87         AcpiOsPrintf ("[NULL NAME]");
  88         return;
  89     }

  91     /* Convert handle to full pathname and print it (with supplied message) */

  93     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;

  95     Status = AcpiNsHandleToPathname (Node, &Buffer);
  96     if (ACPI_SUCCESS (Status))
  97     {
  98         if (Message)
  99         {
 100             AcpiOsPrintf ("%s ", Message);
 101         }

 103         AcpiOsPrintf ("[%s] (Node %p)", (char *) Buffer.Pointer, Node);
 104         ACPI_FREE (Buffer.Pointer);
 105     }
 106 }


 109 /******************************************************************************
 110  *
 115  * FUNCTION:    AcpiNsValidRootPrefix
 116  *
 117  * PARAMETERS:  Prefix          - Character to be checked
 118  *
 119  * RETURN:      TRUE if a valid prefix
 120  *
 121  * DESCRIPTION: Check if a character is a valid ACPI Root prefix
 122  *
 123  *****************************************************************************/
```

```
125 BOOLEAN
126 AcpiNsValidRootPrefix (
127     char                    Prefix)
128 {
130     return ((BOOLEAN) (Prefix == '\\'));
131 }


134 /******************************************************************************
135  *
136  * FUNCTION:    AcpiNsValidPathSeparator
137  *
138  * PARAMETERS:  Sep         - Character to be checked
139  *
140  * RETURN:      TRUE if a valid path separator
141  *
142  * DESCRIPTION: Check if a character is a valid ACPI path separator
143  *
144  *****************************************************************************/

146 static BOOLEAN
147 AcpiNsValidPathSeparator (
148     char                    Sep)
149 {
151     return ((BOOLEAN) (Sep == '.'));
152 }


155 /******************************************************************************
156  *
111  * FUNCTION:    AcpiNsGetType
112  *
113  * PARAMETERS:  Node        - Parent Node to be examined
114  *
115  * RETURN:      Type field from Node whose handle is passed
116  *
117  * DESCRIPTION: Return the type of a Namespace node
118  *
119  *****************************************************************************/

121 ACPI_OBJECT_TYPE
122 AcpiNsGetType (
123     ACPI_NAMESPACE_NODE     *Node)
124 {
125     ACPI_FUNCTION_TRACE (NsGetType);


128     if (!Node)
129     {
130         ACPI_WARNING ((AE_INFO, "Null Node parameter"));
131         return_UINT8 (ACPI_TYPE_ANY);
177         return_UINT32 (ACPI_TYPE_ANY);
132     }

134     return_UINT8 (Node->Type);
180     return_UINT32 ((ACPI_OBJECT_TYPE) Node->Type);
135 }


138 /******************************************************************************
139  *
140  * FUNCTION:    AcpiNsLocal
141  *
142  * PARAMETERS:  Type        - A namespace object type
```



```
143  *
144  * RETURN:      LOCAL if names must be found locally in objects of the
145  *              passed type, 0 if enclosing scopes should be searched
146  *
147  * DESCRIPTION: Returns scope rule for the given object type.
148  *
149  *****************************************************************************/

151 UINT32
152 AcpiNsLocal (
153     ACPI_OBJECT_TYPE        Type)
154 {
155     ACPI_FUNCTION_TRACE (NsLocal);


158     if (!AcpiUtValidObjectType (Type))
159     {
160         /* Type code out of range  */

162         ACPI_WARNING ((AE_INFO, "Invalid Object Type 0x%X", Type));
163         return_UINT32 (ACPI_NS_NORMAL);
164     }

166     return_UINT32 (AcpiGbl_NsProperties[Type] & ACPI_NS_LOCAL);
212     return_UINT32 ((UINT32) AcpiGbl_NsProperties[Type] & ACPI_NS_LOCAL);
167 }


170 /******************************************************************************
171  *
172  * FUNCTION:    AcpiNsGetInternalNameLength
173  *
174  * PARAMETERS:  Info            - Info struct initialized with the
175  *                                external name pointer.
176  *
177  * RETURN:      None
178  *
179  * DESCRIPTION: Calculate the length of the internal (AML) namestring
180  *              corresponding to the external (ASL) namestring.
181  *
182  *****************************************************************************/

184 void
185 AcpiNsGetInternalNameLength (
186     ACPI_NAMESTRING_INFO    *Info)
187 {
188     const char              *NextExternalChar;
189     UINT32                  i;


192     ACPI_FUNCTION_ENTRY ();


195     NextExternalChar = Info->ExternalName;
196     Info->NumCarats = 0;
197     Info->NumSegments = 0;
198     Info->FullyQualified = FALSE;

200     /*
201      * For the internal name, the required length is 4 bytes per segment, plus
202      * 1 each for RootPrefix, MultiNamePrefixOp, segment count, trailing null
203      * (which is not really needed, but no there's harm in putting it there)
204      *
205      * strlen() + 1 covers the first NameSeg, which has no path separator
206      */
207     if (ACPI_IS_ROOT_PREFIX (*NextExternalChar))
```

```
253     if (AcpiNsValidRootPrefix (*NextExternalChar))
208     {
209         Info->FullyQualified = TRUE;
210         NextExternalChar++;

212         /* Skip redundant RootPrefix, like \\_SB.PCI0.SBRG.EC0 */

214         while (ACPI_IS_ROOT_PREFIX (*NextExternalChar))
260         while (AcpiNsValidRootPrefix (*NextExternalChar))
215         {
216             NextExternalChar++;
217         }
218     }
219     else
220     {
221         /* Handle Carat prefixes */

223         while (ACPI_IS_PARENT_PREFIX (*NextExternalChar))
269         while (*NextExternalChar == '^')
224         {
225             Info->NumCarats++;
226             NextExternalChar++;
227         }
228     }

230     /*
231      * Determine the number of ACPI name "segments" by counting the number of
232      * path separators within the string. Start with one segment since the
233      * segment count is [(# separators) + 1], and zero separators is ok.
234      */
235     if (*NextExternalChar)
236     {
237         Info->NumSegments = 1;
238         for (i = 0; NextExternalChar[i]; i++)
239         {
240             if (ACPI_IS_PATH_SEPARATOR (NextExternalChar[i]))
286             if (AcpiNsValidPathSeparator (NextExternalChar[i]))
241             {
242                 Info->NumSegments++;
243             }
244         }
245     }

247     Info->Length = (ACPI_NAME_SIZE * Info->NumSegments) +
248                     4 + Info->NumCarats;

250     Info->NextExternalChar = NextExternalChar;
251 }


254 /*******************************************************************************
255  *
256  * FUNCTION:    AcpiNsBuildInternalName
257  *
258  * PARAMETERS:  Info            - Info struct fully initialized
259  *
260  * RETURN:      Status
261  *
262  * DESCRIPTION: Construct the internal (AML) namestring
263  *              corresponding to the external (ASL) namestring.
264  *
265  ******************************************************************************/

267 ACPI_STATUS
268 AcpiNsBuildInternalName (
269     ACPI_NAMESTRING_INFO    *Info)
```



```
270 {
271     UINT32                  NumSegments = Info->NumSegments;
272     char                    *InternalName = Info->InternalName;
273     const char              *ExternalName = Info->NextExternalChar;
274     char                    *Result = NULL;
275     UINT32                  i;


278     ACPI_FUNCTION_TRACE (NsBuildInternalName);


281     /* Setup the correct prefixes, counts, and pointers */

283     if (Info->FullyQualified)
284     {
285         InternalName[0] = AML_ROOT_PREFIX;
331         InternalName[0] = '\\';

287         if (NumSegments <= 1)
288         {
289             Result = &InternalName[1];
290         }
291         else if (NumSegments == 2)
292         {
293             InternalName[1] = AML_DUAL_NAME_PREFIX;
294             Result = &InternalName[2];
295         }
296         else
297         {
298             InternalName[1] = AML_MULTI_NAME_PREFIX_OP;
299             InternalName[2] = (char) NumSegments;
300             Result = &InternalName[3];
301         }
302     }
303     else
304     {
305         /*
306          * Not fully qualified.
307          * Handle Carats first, then append the name segments
308          */
309         i = 0;
310         if (Info->NumCarats)
311         {
312             for (i = 0; i < Info->NumCarats; i++)
313             {
314                 InternalName[i] = AML_PARENT_PREFIX;
360                 InternalName[i] = '^';
315             }
316         }

318         if (NumSegments <= 1)
319         {
320             Result = &InternalName[i];
321         }
322         else if (NumSegments == 2)
323         {
324             InternalName[i] = AML_DUAL_NAME_PREFIX;
325             Result = &InternalName[(ACPI_SIZE) i+1];
326         }
327         else
328         {
329             InternalName[i] = AML_MULTI_NAME_PREFIX_OP;
330             InternalName[(ACPI_SIZE) i+1] = (char) NumSegments;
331             Result = &InternalName[(ACPI_SIZE) i+2];
332         }
333     }
```

```
335     /* Build the name (minus path separators) */

337     for (; NumSegments; NumSegments--)
338     {
339         for (i = 0; i < ACPI_NAME_SIZE; i++)
340         {
341             if (ACPI_IS_PATH_SEPARATOR (*ExternalName) ||
387             if (AcpiNsValidPathSeparator (*ExternalName) ||
342                 (*ExternalName == 0))
343             {
344                 /* Pad the segment with underscore(s) if segment is short */

346                 Result[i] = '_';
347             }
348             else
349             {
350                 /* Convert the character to uppercase and save it */

352                 Result[i] = (char) ACPI_TOUPPER ((int) *ExternalName);
353                 ExternalName++;
354             }
355         }

357         /* Now we must have a path separator, or the pathname is bad */

359         if (!ACPI_IS_PATH_SEPARATOR (*ExternalName) &&
405         if (!AcpiNsValidPathSeparator (*ExternalName) &&
360             (*ExternalName != 0))
361         {
362             return_ACPI_STATUS (AE_BAD_PATHNAME);
408             return_ACPI_STATUS (AE_BAD_PARAMETER);
363         }

365         /* Move on the next segment */

367         ExternalName++;
368         Result += ACPI_NAME_SIZE;
369     }

371     /* Terminate the string */

373     *Result = 0;

375     if (Info->FullyQualified)
376     {
377         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Returning [%p] (abs) \"\\%s\"\n",
378             InternalName, InternalName));
379     }
380     else
381     {
382         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Returning [%p] (rel) \"%s\"\n",
383             InternalName, InternalName));
384     }

386     return_ACPI_STATUS (AE_OK);
387 }
```

*unchanged_portion_omitted_*

```
453 /*******************************************************************************
454  *
455  * FUNCTION:    AcpiNsExternalizeName
456  *
457  * PARAMETERS:  InternalNameLength  - Lenth of the internal name below
458  *              InternalName        - Internal representation of name
```



```
459  *              ConvertedNameLength - Where the length is returned
460  *              ConvertedName       - Where the resulting external name
461  *                                    is returned
462  *
463  * RETURN:      Status
464  *
465  * DESCRIPTION: Convert internal name (e.g. 5c 2f 02 5f 50 52 5f 43 50 55 30)
466  *              to its external (printable) form (e.g. "\_PR_.CPU0")
467  *
468  ******************************************************************************/

470 ACPI_STATUS
471 AcpiNsExternalizeName (
472     UINT32                  InternalNameLength,
473     const char              *InternalName,
474     UINT32                  *ConvertedNameLength,
475     char                    **ConvertedName)
476 {
477     UINT32                  NamesIndex = 0;
478     UINT32                  NumSegments = 0;
479     UINT32                  RequiredLength;
480     UINT32                  PrefixLength = 0;
481     UINT32                  i = 0;
482     UINT32                  j = 0;


485     ACPI_FUNCTION_TRACE (NsExternalizeName);


488     if (!InternalNameLength     ||
489         !InternalName           ||
490         !ConvertedName)
491     {
492         return_ACPI_STATUS (AE_BAD_PARAMETER);
493     }

495     /* Check for a prefix (one '\' | one or more '^') */

497     switch (InternalName[0])
498     {
499     case AML_ROOT_PREFIX:

545     case '\\':
501         PrefixLength = 1;
502         break;

504     case AML_PARENT_PREFIX:

549     case '^':
506         for (i = 0; i < InternalNameLength; i++)
507         {
508             if (ACPI_IS_PARENT_PREFIX (InternalName[i]))
552             if (InternalName[i] == '^')
509             {
510                 PrefixLength = i + 1;
511             }
512             else
513             {
514                 break;
515             }
516         }

518         if (i == InternalNameLength)
519         {
520             PrefixLength = i;
521         }
```

```
523         break;

525     default:

527         break;
528     }

530     /*
531      * Check for object names. Note that there could be 0-255 of these
532      * 4-byte elements.
533      */
534     if (PrefixLength < InternalNameLength)
535     {
536         switch (InternalName[PrefixLength])
537         {
538         case AML_MULTI_NAME_PREFIX_OP:

540             /* <count> 4-byte names */

542             NamesIndex = PrefixLength + 2;
543             NumSegments = (UINT8)
544                 InternalName[(ACPI_SIZE) PrefixLength + 1];
545             break;

547         case AML_DUAL_NAME_PREFIX:

549             /* Two 4-byte names */

551             NamesIndex = PrefixLength + 1;
552             NumSegments = 2;
553             break;

555         case 0:

557             /* NullName */

559             NamesIndex = 0;
560             NumSegments = 0;
561             break;

563         default:

565             /* one 4-byte name */

567             NamesIndex = PrefixLength;
568             NumSegments = 1;
569             break;
570         }
571     }

573     /*
574      * Calculate the length of ConvertedName, which equals the length
575      * of the prefix, length of all object names, length of any required
576      * punctuation ('.') between object names, plus the NULL terminator.
577      */
578     RequiredLength = PrefixLength + (4 * NumSegments) +
579                         ((NumSegments > 0) ? (NumSegments - 1) : 0) + 1;

581     /*
582      * Check to see if we're still in bounds. If not, there's a problem
583      * with InternalName (invalid format).
584      */
585     if (RequiredLength > InternalNameLength)
586     {
587         ACPI_ERROR ((AE_INFO, "Invalid internal name"));
```



```
588         return_ACPI_STATUS (AE_BAD_PATHNAME);
589     }

591     /* Build the ConvertedName */

593     *ConvertedName = ACPI_ALLOCATE_ZEROED (RequiredLength);
594     if (!(*ConvertedName))
595     {
596         return_ACPI_STATUS (AE_NO_MEMORY);
597     }

599     j = 0;

601     for (i = 0; i < PrefixLength; i++)
602     {
603         (*ConvertedName)[j++] = InternalName[i];
604     }

606     if (NumSegments > 0)
607     {
608         for (i = 0; i < NumSegments; i++)
609         {
610             if (i > 0)
611             {
612                 (*ConvertedName)[j++] = '.';
613             }

615             /* Copy and validate the 4-char name segment */

617             ACPI_MOVE_NAME (&(*ConvertedName)[j], &InternalName[NamesIndex]);
618             AcpiUtRepairName (&(*ConvertedName)[j]);

620             j += ACPI_NAME_SIZE;
621             NamesIndex += ACPI_NAME_SIZE;
658             (*ConvertedName)[j++] = InternalName[NamesIndex++];
659             (*ConvertedName)[j++] = InternalName[NamesIndex++];
660             (*ConvertedName)[j++] = InternalName[NamesIndex++];
661             (*ConvertedName)[j++] = InternalName[NamesIndex++];
622         }
623     }

625     if (ConvertedNameLength)
626     {
627         *ConvertedNameLength = (UINT32) RequiredLength;
628     }

630     return_ACPI_STATUS (AE_OK);
631 }
```

*unchanged_portion_omitted_*

```
680 /*******************************************************************************
681  *
682  * FUNCTION:    AcpiNsTerminate
683  *
684  * PARAMETERS:  none
685  *
686  * RETURN:      none
687  *
688  * DESCRIPTION: free memory allocated for namespace and ACPI table storage.
689  *
690  ******************************************************************************/

692 void
693 AcpiNsTerminate (
694     void)
```

```
695 {
696     ACPI_STATUS             Status;
736     ACPI_OPERAND_OBJECT     *ObjDesc;


699     ACPI_FUNCTION_TRACE (NsTerminate);


702     /*
703      * Free the entire namespace -- all nodes and all objects
704      * attached to the nodes
743      * 1) Free the entire namespace -- all nodes and objects
744      *
745      * Delete all object descriptors attached to namepsace nodes
705      */
706     AcpiNsDeleteNamespaceSubtree (AcpiGbl_RootNode);

708     /* Delete any objects attached to the root node */
749     /* Detach any objects attached to the root */

710     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
711     if (ACPI_FAILURE (Status))
751     ObjDesc = AcpiNsGetAttachedObject (AcpiGbl_RootNode);
752     if (ObjDesc)
712     {
713         return_VOID;
754         AcpiNsDetachObject (AcpiGbl_RootNode);
714     }

716     AcpiNsDeleteNode (AcpiGbl_RootNode);
717     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

719     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Namespace freed\n"));
720     return_VOID;
721 }


724 /*******************************************************************************
725  *
726  * FUNCTION:    AcpiNsOpensScope
727  *
728  * PARAMETERS:  Type        - A valid namespace type
729  *
730  * RETURN:      NEWSCOPE if the passed type "opens a name scope" according
731  *              to the ACPI specification, else 0
732  *
733  ******************************************************************************/

735 UINT32
736 AcpiNsOpensScope (
737     ACPI_OBJECT_TYPE        Type)
738 {
739     ACPI_FUNCTION_ENTRY ();
777     ACPI_FUNCTION_TRACE_STR (NsOpensScope, AcpiUtGetTypeName (Type));


742     if (Type > ACPI_TYPE_LOCAL_MAX)
780     if (!AcpiUtValidObjectType (Type))
743     {
744         /* type code out of range  */

746         ACPI_WARNING ((AE_INFO, "Invalid Object Type 0x%X", Type));
747         return (ACPI_NS_NORMAL);
785         return_UINT32 (ACPI_NS_NORMAL);
748     }
```



```
750     return (((UINT32) AcpiGbl_NsProperties[Type]) & ACPI_NS_NEWSCOPE);
788     return_UINT32 (((UINT32) AcpiGbl_NsProperties[Type]) & ACPI_NS_NEWSCOPE);
751 }


754 /*******************************************************************************
755  *
756  * FUNCTION:    AcpiNsGetNode
757  *
758  * PARAMETERS:  *Pathname   - Name to be found, in external (ASL) format. The
759  *                            \ (backslash) and ^ (carat) prefixes, and the
760  *                            . (period) to separate segments are supported.
761  *              PrefixNode   - Root of subtree to be searched, or NS_ALL for the
762  *                             root of the name space. If Name is fully
763  *                             qualified (first INT8 is '\'), the passed value
764  *                             of Scope will not be accessed.
765  *              Flags       - Used to indicate whether to perform upsearch or
766  *                            not.
767  *              ReturnNode  - Where the Node is returned
768  *
769  * DESCRIPTION: Look up a name relative to a given scope and return the
770  *              corresponding Node. NOTE: Scope can be null.
771  *
772  * MUTEX:       Locks namespace
773  *
774  ******************************************************************************/

776 ACPI_STATUS
777 AcpiNsGetNode (
778     ACPI_NAMESPACE_NODE     *PrefixNode,
779     const char              *Pathname,
780     UINT32                  Flags,
781     ACPI_NAMESPACE_NODE     **ReturnNode)
782 {
783     ACPI_GENERIC_STATE      ScopeInfo;
784     ACPI_STATUS             Status;
785     char                    *InternalPath;


788     ACPI_FUNCTION_TRACE_PTR (NsGetNode, ACPI_CAST_PTR (char, Pathname));


791     /* Simplest case is a null pathname */

793     if (!Pathname)
794     {
795         *ReturnNode = PrefixNode;
796         if (!PrefixNode)
797         {
798             *ReturnNode = AcpiGbl_RootNode;
799         }
800         return_ACPI_STATUS (AE_OK);
801     }

803     /* Quick check for a reference to the root */

805     if (ACPI_IS_ROOT_PREFIX (Pathname[0]) && (!Pathname[1]))
806     {
807         *ReturnNode = AcpiGbl_RootNode;
808         return_ACPI_STATUS (AE_OK);
809     }

811     /* Convert path to internal representation */

813     Status = AcpiNsInternalizeName (Pathname, &InternalPath);
814     if (ACPI_FAILURE (Status))
```

```
 815     {
 816         return_ACPI_STATUS (Status);
 817     }

 819     /* Must lock namespace during lookup */

 821     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
 822     if (ACPI_FAILURE (Status))
 823     {
 824         goto Cleanup;
 825     }

 827     /* Setup lookup scope (search starting point) */

 829     ScopeInfo.Scope.Node = PrefixNode;

 831     /* Lookup the name in the namespace */

 833     Status = AcpiNsLookup (&ScopeInfo, InternalPath, ACPI_TYPE_ANY,
 834                 ACPI_IMODE_EXECUTE, (Flags | ACPI_NS_DONT_OPEN_SCOPE),
 835                 NULL, ReturnNode);
 836     if (ACPI_FAILURE (Status))
 837     {
 838         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "%s, %s\n",
 839                 Pathname, AcpiFormatException (Status)));
 840     }

 842     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

 844 Cleanup:
 845     ACPI_FREE (InternalPath);
 846     return_ACPI_STATUS (Status);
 847 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   12574 Thu Dec 26 13:49:22 2013
new/usr/src/common/acpica/components/namespace/nswalk.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nswalk - Functions for walking the ACPI namespace
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSWALK_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nswalk")


/******************************************************************************
 *
 * FUNCTION:    AcpiNsGetNextNode
 *
```



```
 * PARAMETERS:  ParentNode          - Parent node whose children we are
 *                                    getting
 *              ChildNode           - Previous child that was found.
 *                                    The NEXT child will be returned
 *
 * RETURN:      ACPI_NAMESPACE_NODE - Pointer to the NEXT child or NULL if
 *                                    none is found.
 *
 * DESCRIPTION: Return the next peer node within the namespace. If Handle
 *              is valid, Scope is ignored. Otherwise, the first node
 *              within Scope is returned.
 *
 *****************************************************************************/

ACPI_NAMESPACE_NODE *
AcpiNsGetNextNode (
    ACPI_NAMESPACE_NODE     *ParentNode,
    ACPI_NAMESPACE_NODE     *ChildNode)
{
    ACPI_FUNCTION_ENTRY ();


    if (!ChildNode)
    {
        /* It's really the parent's _scope_ that we want */

        return (ParentNode->Child);
    }

    /* Otherwise just return the next peer */

    return (ChildNode->Peer);
}
```

*unchanged_portion_omitted_*

```
/******************************************************************************
 *
 * FUNCTION:    AcpiNsWalkNamespace
 *
 * PARAMETERS:  Type                - ACPI_OBJECT_TYPE to search for
 *              StartNode           - Handle in namespace where search begins
 *              MaxDepth            - Depth to which search is to reach
 *              Flags               - Whether to unlock the NS before invoking
 *                                    the callback routine
 *              DescendingCallback  - Called during tree descent
 *              PreOrderVisit       - Called during tree pre-order visit
 *                                    when an object of "Type" is found
 *              AscendingCallback   - Called during tree ascent
 *              PostOrderVisit      - Called during tree post-order visit
 *                                    when an object of "Type" is found
 *              Context             - Passed to user function(s) above
 *              ReturnValue         - from the UserFunction if terminated
 *                                    early. Otherwise, returns NULL.
 * RETURNS:     Status
 *
 * DESCRIPTION: Performs a modified depth-first walk of the namespace tree,
 *              starting (and ending) at the node specified by StartHandle.
 *              The callback function is called whenever a node that matches
 *              the type parameter is found. If the callback function returns
 *              a non-zero value, the search is terminated immediately and
 *              this value is returned to the caller.
 *
 *              The point of this procedure is to provide a generic namespace
 *              walk routine that can be called from multiple places to
 *              provide multiple services; the callback function(s) can be
```

```
 187  *              tailored to each task, whether it is a print function,
 188  *              a compare function, etc.
 189  *
 190  ******************************************************************************/

 192 ACPI_STATUS
 193 AcpiNsWalkNamespace (
 194     ACPI_OBJECT_TYPE        Type,
 195     ACPI_HANDLE             StartNode,
 196     UINT32                  MaxDepth,
 197     UINT32                  Flags,
 198     ACPI_WALK_CALLBACK      DescendingCallback,
 199     ACPI_WALK_CALLBACK      AscendingCallback,
 198     ACPI_WALK_CALLBACK      PreOrderVisit,
 199     ACPI_WALK_CALLBACK      PostOrderVisit,
 200     void                    *Context,
 201     void                    **ReturnValue)
 202 {
 203     ACPI_STATUS             Status;
 204     ACPI_STATUS             MutexStatus;
 205     ACPI_NAMESPACE_NODE     *ChildNode;
 206     ACPI_NAMESPACE_NODE     *ParentNode;
 207     ACPI_OBJECT_TYPE        ChildType;
 208     UINT32                  Level;
 209     BOOLEAN                 NodePreviouslyVisited = FALSE;


 212     ACPI_FUNCTION_TRACE (NsWalkNamespace);


 215     /* Special case for the namespace Root Node */

 217     if (StartNode == ACPI_ROOT_OBJECT)
 218     {
 219         StartNode = AcpiGbl_RootNode;
 220     }

 222     /* Null child means "get first node" */

 224     ParentNode  = StartNode;
 225     ChildNode   = AcpiNsGetNextNode (ParentNode, NULL);
 226     ChildType   = ACPI_TYPE_ANY;
 227     Level       = 1;

 229     /*
 230      * Traverse the tree of nodes until we bubble back up to where we
 231      * started. When Level is zero, the loop is done because we have
 232      * bubbled up to (and passed) the original parent handle (StartEntry)
 233      */
 234     while (Level > 0 && ChildNode)
 235     {
 236         Status = AE_OK;

 238         /* Found next child, get the type if we are not searching for ANY */

 240         if (Type != ACPI_TYPE_ANY)
 241         {
 242             ChildType = ChildNode->Type;
 243         }

 245         /*
 246          * Ignore all temporary namespace nodes (created during control
 247          * method execution) unless told otherwise. These temporary nodes
 248          * can cause a race condition because they can be deleted during
 249          * the execution of the user function (if the namespace is
 250          * unlocked before invocation of the user function.) Only the
```



```
 251          * debugger namespace dump will examine the temporary nodes.
 252          */
 253         if ((ChildNode->Flags & ANOBJ_TEMPORARY) &&
 254             !(Flags & ACPI_NS_WALK_TEMP_NODES))
 255         {
 256             Status = AE_CTRL_DEPTH;
 257         }

 259         /* Type must match requested type */

 261         else if (ChildType == Type)
 262         {
 263             /*
 264              * Found a matching node, invoke the user callback function.
 265              * Unlock the namespace if flag is set.
 266              */
 267             if (Flags & ACPI_NS_WALK_UNLOCK)
 268             {
 269                 MutexStatus = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
 270                 if (ACPI_FAILURE (MutexStatus))
 271                 {
 272                     return_ACPI_STATUS (MutexStatus);
 273                 }
 274             }

 276             /*
 277              * Invoke the user function, either descending, ascending,
 277              * Invoke the user function, either pre-order or post-order
 278              * or both.
 279              */
 280             if (!NodePreviouslyVisited)
 281             {
 282                 if (DescendingCallback)
 282                 if (PreOrderVisit)
 283                 {
 284                     Status = DescendingCallback (ChildNode, Level,
 284                     Status = PreOrderVisit (ChildNode, Level,
 285                                 Context, ReturnValue);
 286                 }
 287             }
 288             else
 289             {
 290                 if (AscendingCallback)
 290                 if (PostOrderVisit)
 291                 {
 292                     Status = AscendingCallback (ChildNode, Level,
 292                     Status = PostOrderVisit (ChildNode, Level,
 293                                 Context, ReturnValue);
 294                 }
 295             }

 297             if (Flags & ACPI_NS_WALK_UNLOCK)
 298             {
 299                 MutexStatus = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
 300                 if (ACPI_FAILURE (MutexStatus))
 301                 {
 302                     return_ACPI_STATUS (MutexStatus);
 303                 }
 304             }

 306             switch (Status)
 307             {
 308             case AE_OK:
 309             case AE_CTRL_DEPTH:

 311                 /* Just keep going */
```

```
312                 break;

314             case AE_CTRL_TERMINATE:

316                 /* Exit now, with OK status */

318                 return_ACPI_STATUS (AE_OK);

320             default:

322                 /* All others are valid exceptions */

324                 return_ACPI_STATUS (Status);
325             }
326         }

328         /*
329          * Depth first search: Attempt to go down another level in the
330          * namespace if we are allowed to. Don't go any further if we have
331          * reached the caller specified maximum depth or if the user
332          * function has specified that the maximum depth has been reached.
333          */
334         if (!NodePreviouslyVisited &&
335             (Level < MaxDepth) &&
336             (Status != AE_CTRL_DEPTH))
337         {
338             if (ChildNode->Child)
339             {
340                 /* There is at least one child of this node, visit it */

342                 Level++;
343                 ParentNode = ChildNode;
344                 ChildNode = AcpiNsGetNextNode (ParentNode, NULL);
345                 continue;
346             }
347         }

349         /* No more children, re-visit this node */

351         if (!NodePreviouslyVisited)
352         {
353             NodePreviouslyVisited = TRUE;
354             continue;
355         }

357         /* No more children, visit peers */

359         ChildNode = AcpiNsGetNextNode (ParentNode, ChildNode);
360         if (ChildNode)
361         {
362             NodePreviouslyVisited = FALSE;
363         }

365         /* No peers, re-visit parent */

367         else
368         {
369             /*
370              * No more children of this node (AcpiNsGetNextNode failed), go
371              * back upwards in the namespace tree to the node's parent.
372              */
373             Level--;
374             ChildNode = ParentNode;
375             ParentNode = ParentNode->Parent;

377             NodePreviouslyVisited = TRUE;
```



```
378         }
379     }

381     /* Complete walk, not terminated by user function */

383     return_ACPI_STATUS (AE_OK);
384 }
```

```
**********************************************************
   32410 Thu Dec 26 13:49:22 2013
new/usr/src/common/acpica/components/namespace/nsxfeval.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: nsxfeval - Public interfaces to the ACPI subsystem
 *                         ACPI Object evaluation interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSXFEVAL_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsxfeval")
```



```
/* Local prototypes */

static void
AcpiNsResolveReferences (
    ACPI_EVALUATE_INFO      *Info);


/*******************************************************************************
 *
 * FUNCTION:    AcpiEvaluateObjectTyped
 *
 * PARAMETERS:  Handle              - Object handle (optional)
 *              Pathname            - Object pathname (optional)
 *              ExternalParams      - List of parameters to pass to method,
 *                                    terminated by NULL. May be NULL
 *                                    if no parameters are being passed.
 *              ReturnBuffer        - Where to put method's return value (if
 *                                    any). If NULL, no value is returned.
 *              ReturnType          - Expected type of return object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Find and evaluate the given object, passing the given
 *              parameters if necessary. One of "Handle" or "Pathname" must
 *              be valid (non-null)
 *
 ******************************************************************************/

ACPI_STATUS
AcpiEvaluateObjectTyped (
    ACPI_HANDLE             Handle,
    ACPI_STRING             Pathname,
    ACPI_OBJECT_LIST        *ExternalParams,
    ACPI_BUFFER             *ReturnBuffer,
    ACPI_OBJECT_TYPE        ReturnType)
{
    ACPI_STATUS             Status;
    BOOLEAN                 FreeBufferOnError = FALSE;
    BOOLEAN                 MustFree = FALSE;


    ACPI_FUNCTION_TRACE (AcpiEvaluateObjectTyped);


    /* Return buffer must be valid */

    if (!ReturnBuffer)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (ReturnBuffer->Length == ACPI_ALLOCATE_BUFFER)
    {
        FreeBufferOnError = TRUE;
        MustFree = TRUE;
    }

    /* Evaluate the object */

    Status = AcpiEvaluateObject (Handle, Pathname,
        ExternalParams, ReturnBuffer);
    Status = AcpiEvaluateObject (Handle, Pathname, ExternalParams, ReturnBuffer)
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }
```

```
122     /* Type ANY means "don't care" */

124     if (ReturnType == ACPI_TYPE_ANY)
125     {
126         return_ACPI_STATUS (AE_OK);
127     }

129     if (ReturnBuffer->Length == 0)
130     {
131         /* Error because caller specifically asked for a return value */

133         ACPI_ERROR ((AE_INFO, "No return value"));
134         return_ACPI_STATUS (AE_NULL_OBJECT);
135     }

137     /* Examine the object type returned from EvaluateObject */

139     if (((ACPI_OBJECT *) ReturnBuffer->Pointer)->Type == ReturnType)
140     {
141         return_ACPI_STATUS (AE_OK);
142     }

144     /* Return object type does not match requested type */

146     ACPI_ERROR ((AE_INFO,
147         "Incorrect return type [%s] requested [%s]",
148         AcpiUtGetTypeName (((ACPI_OBJECT *) ReturnBuffer->Pointer)->Type),
149         AcpiUtGetTypeName (ReturnType)));

151     if (FreeBufferOnError)
149     if (MustFree)
152     {
153         /*
154          * Free a buffer created via ACPI_ALLOCATE_BUFFER.
155          * Note: We use AcpiOsFree here because AcpiOsAllocate was used
156          * to allocate the buffer. This purposefully bypasses the
157          * (optionally enabled) allocation tracking mechanism since we
158          * only want to track internal allocations.
159          */
151         /* Caller used ACPI_ALLOCATE_BUFFER, free the return buffer */

160         AcpiOsFree (ReturnBuffer->Pointer);
161         ReturnBuffer->Pointer = NULL;
162     }

164     ReturnBuffer->Length = 0;
165     return_ACPI_STATUS (AE_TYPE);
166 }

168 ACPI_EXPORT_SYMBOL (AcpiEvaluateObjectTyped)


171 /*******************************************************************************
172  *
173  * FUNCTION:    AcpiEvaluateObject
174  *
175  * PARAMETERS:  Handle              - Object handle (optional)
176  *              Pathname            - Object pathname (optional)
177  *              ExternalParams      - List of parameters to pass to method,
178  *                                    terminated by NULL. May be NULL
179  *                                    if no parameters are being passed.
180  *              ReturnBuffer        - Where to put method's return value (if
181  *                                    any). If NULL, no value is returned.
182  *
183  * RETURN:      Status
```



```
184  *
185  * DESCRIPTION: Find and evaluate the given object, passing the given
186  *              parameters if necessary. One of "Handle" or "Pathname" must
187  *              be valid (non-null)
188  *
189  ******************************************************************************/

191 ACPI_STATUS
192 AcpiEvaluateObject (
193     ACPI_HANDLE             Handle,
194     ACPI_STRING             Pathname,
195     ACPI_OBJECT_LIST        *ExternalParams,
196     ACPI_BUFFER             *ReturnBuffer)
197 {
198     ACPI_STATUS             Status;
199     ACPI_EVALUATE_INFO      *Info;
200     ACPI_SIZE               BufferSpaceNeeded;
201     UINT32                  i;


204     ACPI_FUNCTION_TRACE (AcpiEvaluateObject);


207     /* Allocate and initialize the evaluation information block */

209     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
210     if (!Info)
211     {
212         return_ACPI_STATUS (AE_NO_MEMORY);
213     }

208     Info->Pathname = Pathname;

215     /* Convert and validate the device handle */

217     Info->PrefixNode = AcpiNsValidateHandle (Handle);
218     if (!Info->PrefixNode)
219     {
220         Status = AE_BAD_PARAMETER;
221         goto Cleanup;
222     }

224     /*
225      * Get the actual namespace node for the target object.
226      * Handles these cases:
227      *
228      * 1) Null node, valid pathname from root (absolute path)
229      * 2) Node and valid pathname (path relative to Node)
230      * 3) Node, Null pathname
220      * If there are parameters to be passed to a control method, the external
221      * objects must all be converted to internal objects
231      */
232     if ((Pathname) &&
233         (ACPI_IS_ROOT_PREFIX (Pathname[0])))
234     {
235         /* The path is fully qualified, just evaluate by name */

237         Info->PrefixNode = NULL;
238     }
239     else if (!Handle)
240     {
241         /*
242          * A handle is optional iff a fully qualified pathname is specified.
243          * Since we've already handled fully qualified names above, this is
244          * an error.
245          */
```

```
        if (!Pathname)
        {
            ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
                "Both Handle and Pathname are NULL"));
        }
        else
        {
            ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
                "Null Handle with relative pathname [%s]", Pathname));
        }

        Status = AE_BAD_PARAMETER;
        goto Cleanup;
    }

    Info->RelativePathname = Pathname;

    /*
     * Convert all external objects passed as arguments to the
     * internal version(s).
     */
    if (ExternalParams && ExternalParams->Count)
    {
        Info->ParamCount = (UINT16) ExternalParams->Count;

        /* Warn on impossible argument count */

        if (Info->ParamCount > ACPI_METHOD_NUM_ARGS)
        {
            ACPI_WARN_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
                "Excess arguments (%u) - using only %u",
                Info->ParamCount, ACPI_METHOD_NUM_ARGS));

            Info->ParamCount = ACPI_METHOD_NUM_ARGS;
        }

        /*
         * Allocate a new parameter block for the internal objects
         * Add 1 to count to allow for null terminated internal list
         */
        Info->Parameters = ACPI_ALLOCATE_ZEROED (
            ((ACPI_SIZE) Info->ParamCount + 1) * sizeof (void *));
        ((ACPI_SIZE) ExternalParams->Count + 1) * sizeof (void *));
        if (!Info->Parameters)
        {
            Status = AE_NO_MEMORY;
            goto Cleanup;
        }

        /* Convert each external object in the list to an internal object */

        for (i = 0; i < Info->ParamCount; i++)
        for (i = 0; i < ExternalParams->Count; i++)
        {
            Status = AcpiUtCopyEobjectToIobject (
                &ExternalParams->Pointer[i], &Info->Parameters[i]);
            if (ACPI_FAILURE (Status))
            {
                goto Cleanup;
            }
        }

        Info->Parameters[Info->ParamCount] = NULL;
        Info->Parameters[ExternalParams->Count] = NULL;
    }
```



```
#if 0

    /*
     * Begin incoming argument count analysis. Check for too few args
     * and too many args.
     * Three major cases:
     * 1) Fully qualified pathname
     * 2) No handle, not fully qualified pathname (error)
     * 3) Valid handle
     */

    switch (AcpiNsGetType (Info->Node))
    if ((Pathname) &&
        (AcpiNsValidRootPrefix (Pathname[0])))
    {
    case ACPI_TYPE_METHOD:
        /* The path is fully qualified, just evaluate by name */

        /* Check incoming argument count against the method definition */

        if (Info->ObjDesc->Method.ParamCount > Info->ParamCount)
        {
            ACPI_ERROR ((AE_INFO,
                "Insufficient arguments (%u) - %u are required",
                Info->ParamCount,
                Info->ObjDesc->Method.ParamCount));

            Status = AE_MISSING_ARGUMENTS;
            goto Cleanup;
        Info->PrefixNode = NULL;
        Status = AcpiNsEvaluate (Info);
        }

        else if (Info->ObjDesc->Method.ParamCount < Info->ParamCount)
    else if (!Handle)
        {
            ACPI_WARNING ((AE_INFO,
                "Excess arguments (%u) - only %u are required",
                Info->ParamCount,
                Info->ObjDesc->Method.ParamCount));

            /* Just pass the required number of arguments */

            Info->ParamCount = Info->ObjDesc->Method.ParamCount;
        }

        /*
         * Any incoming external objects to be passed as arguments to the
         * method must be converted to internal objects
         * A handle is optional iff a fully qualified pathname is specified.
         * Since we've already handled fully qualified names above, this is
         * an error
         */
        if (Info->ParamCount)
        if (!Pathname)
        {
            /*
             * Allocate a new parameter block for the internal objects
             * Add 1 to count to allow for null terminated internal list
             */
            Info->Parameters = ACPI_ALLOCATE_ZEROED (
                ((ACPI_SIZE) Info->ParamCount + 1) * sizeof (void *));
            if (!Info->Parameters)
            {
                Status = AE_NO_MEMORY;
```

```
361                 goto Cleanup;
274             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
275                 "Both Handle and Pathname are NULL"));
362             }

364             /* Convert each external object in the list to an internal object */

366             for (i = 0; i < Info->ParamCount; i++)
277         else
367             {
368                 Status = AcpiUtCopyEobjectToIobject (
369                     &ExternalParams->Pointer[i], &Info->Parameters[i]);
370                 if (ACPI_FAILURE (Status))
371                 {
372                     goto Cleanup;
279             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
280                 "Null Handle with relative pathname [%s]", Pathname));
373                 }
374             }

376             Info->Parameters[Info->ParamCount] = NULL;
283         Status = AE_BAD_PARAMETER;
377         }
378         break;

380     default:

382         /* Warn if arguments passed to an object that is not a method */

384         if (Info->ParamCount)
285     else
385         {
386             ACPI_WARNING ((AE_INFO,
387                 "%u arguments were passed to a non-method ACPI object",
388                 Info->ParamCount));
389         }
390         break;
391     }
287         /* We have a namespace a node and a possible relative path */

393 #endif


396     /* Now we can evaluate the object */

398     Status = AcpiNsEvaluate (Info);
290     }

400     /*
401      * If we are expecting a return value, and all went well above,
402      * copy the return value to an external object.
403      */
404     if (ReturnBuffer)
405     {
406         if (!Info->ReturnObject)
407         {
408             ReturnBuffer->Length = 0;
409         }
410         else
411         {
412             if (ACPI_GET_DESCRIPTOR_TYPE (Info->ReturnObject) ==
413                 ACPI_DESC_TYPE_NAMED)
414             {
415                 /*
416                  * If we received a NS Node as a return object, this means that
417                  * the object we are evaluating has nothing interesting to
```



```
418                  * return (such as a mutex, etc.)  We return an error because
419                  * these types are essentially unsupported by this interface.
420                  * We don't check up front because this makes it easier to add
421                  * support for various types at a later date if necessary.
422                  */
423                 Status = AE_TYPE;
424                 Info->ReturnObject = NULL;   /* No need to delete a NS Node */
425                 ReturnBuffer->Length = 0;
426             }

428             if (ACPI_SUCCESS (Status))
429             {
430                 /* Dereference Index and RefOf references */

432                 AcpiNsResolveReferences (Info);

434                 /* Get the size of the returned object */

436                 Status = AcpiUtGetObjectSize (Info->ReturnObject,
437                             &BufferSpaceNeeded);
438                 if (ACPI_SUCCESS (Status))
439                 {
440                     /* Validate/Allocate/Clear caller buffer */

442                     Status = AcpiUtInitializeBuffer (ReturnBuffer,
443                                 BufferSpaceNeeded);
444                     if (ACPI_FAILURE (Status))
445                     {
446                         /*
447                          * Caller's buffer is too small or a new one can't
448                          * be allocated
449                          */
450                         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
451                             "Needed buffer size %X, %s\n",
452                             (UINT32) BufferSpaceNeeded,
453                             AcpiFormatException (Status)));
454                     }
455                     else
456                     {
457                         /* We have enough space for the object, build it */

459                         Status = AcpiUtCopyIobjectToEobject (Info->ReturnObject,
460                                     ReturnBuffer);
461                     }
462                 }
463             }
464         }
465     }

467     if (Info->ReturnObject)
468     {
469         /*
470          * Delete the internal return object. NOTE: Interpreter must be
471          * locked to avoid race condition.
472          */
473         AcpiExEnterInterpreter ();

475         /* Remove one reference on the return object (should delete it) */

477         AcpiUtRemoveReference (Info->ReturnObject);
478         AcpiExExitInterpreter ();
479     }


482 Cleanup:
```

```
484     /* Free the input parameter list (if we created one) */

486     if (Info->Parameters)
487     {
488         /* Free the allocated parameter block */

490         AcpiUtDeleteInternalObjectList (Info->Parameters);
491     }

493     ACPI_FREE (Info);
494     return_ACPI_STATUS (Status);
495 }

497 ACPI_EXPORT_SYMBOL (AcpiEvaluateObject)


500 /*******************************************************************************
501  *
502  * FUNCTION:    AcpiNsResolveReferences
503  *
504  * PARAMETERS:  Info                    - Evaluation info block
505  *
506  * RETURN:      Info->ReturnObject is replaced with the dereferenced object
507  *
508  * DESCRIPTION: Dereference certain reference objects. Called before an
509  *              internal return object is converted to an external ACPI_OBJECT.
510  *
511  * Performs an automatic dereference of Index and RefOf reference objects.
512  * These reference objects are not supported by the ACPI_OBJECT, so this is a
513  * last resort effort to return something useful. Also, provides compatibility
514  * with other ACPI implementations.
515  *
516  * NOTE: does not handle references within returned package objects or nested
517  * references, but this support could be added later if found to be necessary.
518  *
519  ******************************************************************************/

521 static void
522 AcpiNsResolveReferences (
523     ACPI_EVALUATE_INFO      *Info)
524 {
525     ACPI_OPERAND_OBJECT     *ObjDesc = NULL;
526     ACPI_NAMESPACE_NODE     *Node;


529     /* We are interested in reference objects only */

531     if ((Info->ReturnObject)->Common.Type != ACPI_TYPE_LOCAL_REFERENCE)
532     {
533         return;
534     }

536     /*
537      * Two types of references are supported - those created by Index and
538      * RefOf operators. A name reference (AML_NAMEPATH_OP) can be converted
539      * to an ACPI_OBJECT, so it is not dereferenced here. A DdbHandle
540      * (AML_LOAD_OP) cannot be dereferenced, nor can it be converted to
541      * an ACPI_OBJECT.
542      */
543     switch (Info->ReturnObject->Reference.Class)
544     {
545     case ACPI_REFCLASS_INDEX:

547         ObjDesc = *(Info->ReturnObject->Reference.Where);
548         break;
```



```
550     case ACPI_REFCLASS_REFOF:

552         Node = Info->ReturnObject->Reference.Object;
553         if (Node)
554         {
555             ObjDesc = Node->Object;
556         }
557         break;

559     default:

561         return;
562     }

564     /* Replace the existing reference object */

566     if (ObjDesc)
567     {
568         AcpiUtAddReference (ObjDesc);
569         AcpiUtRemoveReference (Info->ReturnObject);
570         Info->ReturnObject = ObjDesc;
571     }

573     return;
574 }


577 /*******************************************************************************
578  *
579  * FUNCTION:    AcpiWalkNamespace
580  *
581  * PARAMETERS:  Type                - ACPI_OBJECT_TYPE to search for
582  *              StartObject         - Handle in namespace where search begins
583  *              MaxDepth            - Depth to which search is to reach
584  *              DescendingCallback  - Called during tree descent
475  *              PreOrderVisit       - Called during tree pre-order visit
585  *                                    when an object of "Type" is found
586  *              AscendingCallback   - Called during tree ascent
477  *              PostOrderVisit      - Called during tree post-order visit
587  *                                    when an object of "Type" is found
588  *              Context             - Passed to user function(s) above
589  *              ReturnValue         - Location where return value of
590  *                                    UserFunction is put if terminated early
591  *
592  * RETURNS      Return value from the UserFunction if terminated early.
593  *              Otherwise, returns NULL.
594  *
595  * DESCRIPTION: Performs a modified depth-first walk of the namespace tree,
596  *              starting (and ending) at the object specified by StartHandle.
597  *              The callback function is called whenever an object that matches
598  *              the type parameter is found. If the callback function returns
599  *              a non-zero value, the search is terminated immediately and this
600  *              value is returned to the caller.
601  *
602  *              The point of this procedure is to provide a generic namespace
603  *              walk routine that can be called from multiple places to
604  *              provide multiple services; the callback function(s) can be
605  *              tailored to each task, whether it is a print function,
606  *              a compare function, etc.
607  *
608  ******************************************************************************/

610 ACPI_STATUS
611 AcpiWalkNamespace (
612     ACPI_OBJECT_TYPE        Type,
613     ACPI_HANDLE             StartObject,
```

```
614     UINT32                  MaxDepth,
615     ACPI_WALK_CALLBACK      DescendingCallback,
616     ACPI_WALK_CALLBACK      AscendingCallback,
506     ACPI_WALK_CALLBACK      PreOrderVisit,
507     ACPI_WALK_CALLBACK      PostOrderVisit,
617     void                    *Context,
618     void                    **ReturnValue)
619 {
620     ACPI_STATUS             Status;


623     ACPI_FUNCTION_TRACE (AcpiWalkNamespace);


626     /* Parameter validation */

628     if ((Type > ACPI_TYPE_LOCAL_MAX) ||
629         (!MaxDepth)                  ||
630         (!DescendingCallback && !AscendingCallback))
521         (!PreOrderVisit && !PostOrderVisit))
631     {
632         return_ACPI_STATUS (AE_BAD_PARAMETER);
633     }

635     /*
636      * Need to acquire the namespace reader lock to prevent interference
637      * with any concurrent table unloads (which causes the deletion of
638      * namespace objects). We cannot allow the deletion of a namespace node
639      * while the user function is using it. The exception to this are the
640      * nodes created and deleted during control method execution -- these
641      * nodes are marked as temporary nodes and are ignored by the namespace
642      * walk. Thus, control methods can be executed while holding the
643      * namespace deletion lock (and the user function can execute control
644      * methods.)
645      */
646     Status = AcpiUtAcquireReadLock (&AcpiGbl_NamespaceRwLock);
647     if (ACPI_FAILURE (Status))
648     {
649         return_ACPI_STATUS (Status);
540         return (Status);
650     }

652     /*
653      * Lock the namespace around the walk. The namespace will be
654      * unlocked/locked around each call to the user function - since the user
655      * function must be allowed to make ACPICA calls itself (for example, it
656      * will typically execute control methods during device enumeration.)
657      */
658     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
659     if (ACPI_FAILURE (Status))
660     {
661         goto UnlockAndExit;
662     }

664     /* Now we can validate the starting node */

666     if (!AcpiNsValidateHandle (StartObject))
667     {
668         Status = AE_BAD_PARAMETER;
669         goto UnlockAndExit2;
670     }

672     Status = AcpiNsWalkNamespace (Type, StartObject, MaxDepth,
673                 ACPI_NS_WALK_UNLOCK, DescendingCallback,
674                 AscendingCallback, Context, ReturnValue);
556                 ACPI_NS_WALK_UNLOCK, PreOrderVisit,
```



```
557                 PostOrderVisit, Context, ReturnValue);

676 UnlockAndExit2:
677     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

679 UnlockAndExit:
680     (void) AcpiUtReleaseReadLock (&AcpiGbl_NamespaceRwLock);
681     return_ACPI_STATUS (Status);
682 }

684 ACPI_EXPORT_SYMBOL (AcpiWalkNamespace)


687 /*******************************************************************************
688  *
689  * FUNCTION:    AcpiNsGetDeviceCallback
690  *
691  * PARAMETERS:  Callback from AcpiGetDevice
692  *
693  * RETURN:      Status
694  *
695  * DESCRIPTION: Takes callbacks from WalkNamespace and filters out all non-
696  *              present devices, or if they specified a HID, it filters based
697  *              on that.
698  *
699  ******************************************************************************/

701 static ACPI_STATUS
702 AcpiNsGetDeviceCallback (
703     ACPI_HANDLE             ObjHandle,
704     UINT32                  NestingLevel,
705     void                    *Context,
706     void                    **ReturnValue)
707 {
708     ACPI_GET_DEVICES_INFO   *Info = Context;
709     ACPI_STATUS             Status;
710     ACPI_NAMESPACE_NODE     *Node;
711     UINT32                  Flags;
712     ACPI_PNP_DEVICE_ID      *Hid;
713     ACPI_PNP_DEVICE_ID_LIST *Cid;
594     ACPI_DEVICE_ID          *Hid;
595     ACPI_DEVICE_ID_LIST     *Cid;
714     UINT32                  i;
715     BOOLEAN                 Found;
716     int                     NoMatch;


719     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
720     if (ACPI_FAILURE (Status))
721     {
722         return (Status);
723     }

725     Node = AcpiNsValidateHandle (ObjHandle);
726     Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
727     if (ACPI_FAILURE (Status))
728     {
729         return (Status);
730     }

732     if (!Node)
733     {
734         return (AE_BAD_PARAMETER);
735     }

737     /*
```

```
738      * First, filter based on the device HID and CID.
739      *
740      * 01/2010: For this case where a specific HID is requested, we don't
741      * want to run _STA until we have an actual HID match. Thus, we will
742      * not unnecessarily execute _STA on devices for which the caller
743      * doesn't care about. Previously, _STA was executed unconditionally
744      * on all devices found here.
745      *
746      * A side-effect of this change is that now we will continue to search
747      * for a matching HID even under device trees where the parent device
748      * would have returned a _STA that indicates it is not present or
749      * not functioning (thus aborting the search on that branch).
750      */
751     if (Info->Hid != NULL)
752     {
753         Status = AcpiUtExecute_HID (Node, &Hid);
754         if (Status == AE_NOT_FOUND)
755         {
756             return (AE_OK);
757         }
758         else if (ACPI_FAILURE (Status))
759         {
760             return (AE_CTRL_DEPTH);
761         }
763         NoMatch = ACPI_STRCMP (Hid->String, Info->Hid);
764         ACPI_FREE (Hid);
766         if (NoMatch)
767         {
768             /*
769              * HID does not match, attempt match within the
770              * list of Compatible IDs (CIDs)
771              */
772             Status = AcpiUtExecute_CID (Node, &Cid);
773             if (Status == AE_NOT_FOUND)
774             {
775                 return (AE_OK);
776             }
777             else if (ACPI_FAILURE (Status))
778             {
779                 return (AE_CTRL_DEPTH);
780             }
782             /* Walk the CID list */
784             Found = FALSE;
785             for (i = 0; i < Cid->Count; i++)
786             {
787                 if (ACPI_STRCMP (Cid->Ids[i].String, Info->Hid) == 0)
788                 {
789                     /* Found a matching CID */
791                     Found = TRUE;
792                     break;
793                 }
794             }
796             ACPI_FREE (Cid);
797             if (!Found)
798             {
799                 return (AE_OK);
800             }
801         }
802     }
```



```
804     /* Run _STA to determine if device is present */
806     Status = AcpiUtExecute_STA (Node, &Flags);
807     if (ACPI_FAILURE (Status))
808     {
809         return (AE_CTRL_DEPTH);
810     }
812     if (!(Flags & ACPI_STA_DEVICE_PRESENT) &&
813         !(Flags & ACPI_STA_DEVICE_FUNCTIONING))
814     {
815         /*
816          * Don't examine the children of the device only when the
817          * device is neither present nor functional. See ACPI spec,
818          * description of _STA for more information.
819          */
820         return (AE_CTRL_DEPTH);
821     }
823     /* We have a valid device, invoke the user function */
825     Status = Info->UserFunction (ObjHandle, NestingLevel, Info->Context,
826                 ReturnValue);
827     return (Status);
828 }
```

_____***unchanged_portion_omitted_***

```
1076 ACPI_EXPORT_SYMBOL (AcpiGetData)
```

```
**********************************************************
   20599 Thu Dec 26 13:49:23 2013
new/usr/src/common/acpica/components/namespace/nsxfname.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: nsxfname - Public interfaces to the ACPI subsystem
   4  *                         ACPI Namespace oriented interfaces
   5  *
   6  *****************************************************************************/

   8 /*
   9  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * Copyright (C) 2000 - 2011, Intel Corp.
  10  * All rights reserved.
  11  *
  12  * Redistribution and use in source and binary forms, with or without
  13  * modification, are permitted provided that the following conditions
  14  * are met:
  15  * 1. Redistributions of source code must retain the above copyright
  16  *    notice, this list of conditions, and the following disclaimer,
  17  *    without modification.
  18  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  19  *    substantially similar to the "NO WARRANTY" disclaimer below
  20  *    ("Disclaimer") and any redistribution must be conditioned upon
  21  *    including a substantially similar Disclaimer requirement for further
  22  *    binary redistribution.
  23  * 3. Neither the names of the above-listed copyright holders nor the names
  24  *    of any contributors may be used to endorse or promote products derived
  25  *    from this software without specific prior written permission.
  26  *
  27  * Alternatively, this software may be distributed under the terms of the
  28  * GNU General Public License ("GPL") version 2 as published by the Free
  29  * Software Foundation.
  30  *
  31  * NO WARRANTY
  32  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  33  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  34  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  35  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  36  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  37  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  38  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  39  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  40  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  41  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  42  * POSSIBILITY OF SUCH DAMAGES.
  43  */

  45 #define __NSXFNAME_C__
  46 #define EXPORT_ACPI_INTERFACES

  48 #include "acpi.h"
  49 #include "accommon.h"
  50 #include "acnamesp.h"
  51 #include "acparser.h"
  52 #include "amlcode.h"


  55 #define _COMPONENT          ACPI_NAMESPACE
  56          ACPI_MODULE_NAME    ("nsxfname")

  58 /* Local prototypes */
```



```
  60 static char *
  61 AcpiNsCopyDeviceId (
  62     ACPI_PNP_DEVICE_ID      *Dest,
  63     ACPI_PNP_DEVICE_ID      *Source,
  61     ACPI_DEVICE_ID          *Dest,
  62     ACPI_DEVICE_ID          *Source,
  64     char                    *StringArea);


  67 /******************************************************************************
  68  *
  69  * FUNCTION:    AcpiGetHandle
  70  *
  71  * PARAMETERS:  Parent          - Object to search under (search scope).
  72  *              Pathname        - Pointer to an asciiz string containing the
  73  *                                name
  74  *              RetHandle       - Where the return handle is returned
  75  *
  76  * RETURN:      Status
  77  *
  78  * DESCRIPTION: This routine will search for a caller specified name in the
  79  *              name space. The caller can restrict the search region by
  80  *              specifying a non NULL parent. The parent value is itself a
  81  *              namespace handle.
  82  *
  83  *****************************************************************************/

  85 ACPI_STATUS
  86 AcpiGetHandle (
  87     ACPI_HANDLE             Parent,
  88     ACPI_STRING             Pathname,
  89     ACPI_HANDLE             *RetHandle)
  90 {
  91     ACPI_STATUS             Status;
  92     ACPI_NAMESPACE_NODE     *Node = NULL;
  93     ACPI_NAMESPACE_NODE     *PrefixNode = NULL;


  96     ACPI_FUNCTION_ENTRY ();


  99     /* Parameter Validation */

 101     if (!RetHandle || !Pathname)
 102     {
 103         return (AE_BAD_PARAMETER);
 104     }

 106     /* Convert a parent handle to a prefix node */

 108     if (Parent)
 109     {
 110         PrefixNode = AcpiNsValidateHandle (Parent);
 111         if (!PrefixNode)
 112         {
 113             return (AE_BAD_PARAMETER);
 114         }
 115     }

 117     /*
 118      * Valid cases are:
 119      * 1) Fully qualified pathname
 120      * 2) Parent + Relative pathname
 121      *
 122      * Error for <null Parent + relative path>
```

```
123      */
124     if (ACPI_IS_ROOT_PREFIX (Pathname[0]))
123     if (AcpiNsValidRootPrefix (Pathname[0]))
125     {
126         /* Pathname is fully qualified (starts with '\') */

128         /* Special case for root-only, since we can't search for it */

130         if (!ACPI_STRCMP (Pathname, ACPI_NS_ROOT_PATH))
131         {
132             *RetHandle = ACPI_CAST_PTR (ACPI_HANDLE, AcpiGbl_RootNode);
133             return (AE_OK);
134         }
135     }
136     else if (!PrefixNode)
137     {
138         /* Relative path with null prefix is disallowed */

140         return (AE_BAD_PARAMETER);
141     }

143     /* Find the Node and convert to a handle */

145     Status = AcpiNsGetNode (PrefixNode, Pathname, ACPI_NS_NO_UPSEARCH, &Node);
146     if (ACPI_SUCCESS (Status))
147     {
148         *RetHandle = ACPI_CAST_PTR (ACPI_HANDLE, Node);
149     }

151     return (Status);
152 }

154 ACPI_EXPORT_SYMBOL (AcpiGetHandle)


157 /*******************************************************************************
158  *
159  * FUNCTION:    AcpiGetName
160  *
161  * PARAMETERS:  Handle          - Handle to be converted to a pathname
162  *              NameType        - Full pathname or single segment
163  *              Buffer          - Buffer for returned path
164  *
165  * RETURN:      Pointer to a string containing the fully qualified Name.
166  *
167  * DESCRIPTION: This routine returns the fully qualified name associated with
168  *              the Handle parameter. This and the AcpiPathnameToHandle are
169  *              complementary functions.
170  *
171  ******************************************************************************/

173 ACPI_STATUS
174 AcpiGetName (
175     ACPI_HANDLE             Handle,
176     UINT32                  NameType,
177     ACPI_BUFFER             *Buffer)
178 {
179     ACPI_STATUS             Status;
180     ACPI_NAMESPACE_NODE     *Node;
181     char                    *NodeName;


184     /* Parameter validation */

186     if (NameType > ACPI_NAME_TYPE_MAX)
187     {
```



```
188         return (AE_BAD_PARAMETER);
189     }

191     Status = AcpiUtValidateBuffer (Buffer);
192     if (ACPI_FAILURE (Status))
193     {
194         return (Status);
195     }

197     if (NameType == ACPI_FULL_PATHNAME)
198     {
199         /* Get the full pathname (From the namespace root) */

201         Status = AcpiNsHandleToPathname (Handle, Buffer);
202         return (Status);
203     }

205     /*
206      * Wants the single segment ACPI name.
207      * Validate handle and convert to a namespace Node
208      */
209     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
210     if (ACPI_FAILURE (Status))
211     {
212         return (Status);
213     }

215     Node = AcpiNsValidateHandle (Handle);
216     if (!Node)
217     {
218         Status = AE_BAD_PARAMETER;
219         goto UnlockAndExit;
220     }

222     /* Validate/Allocate/Clear caller buffer */

224     Status = AcpiUtInitializeBuffer (Buffer, ACPI_PATH_SEGMENT_LENGTH);
225     if (ACPI_FAILURE (Status))
226     {
227         goto UnlockAndExit;
228     }

230     /* Just copy the ACPI name from the Node and zero terminate it */

232     NodeName = AcpiUtGetNodeName (Node);
233     ACPI_MOVE_NAME (Buffer->Pointer, NodeName);
230     ACPI_STRNCPY (Buffer->Pointer, AcpiUtGetNodeName (Node),
231                 ACPI_NAME_SIZE);
234     ((char *) Buffer->Pointer) [ACPI_NAME_SIZE] = 0;
235     Status = AE_OK;


238 UnlockAndExit:

240     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
241     return (Status);
242 }

244 ACPI_EXPORT_SYMBOL (AcpiGetName)


247 /*******************************************************************************
248  *
249  * FUNCTION:    AcpiNsCopyDeviceId
250  *
251  * PARAMETERS:  Dest                - Pointer to the destination PNP_DEVICE_ID
```

```
 *              Source              - Pointer to the source PNP_DEVICE_ID
 * PARAMETERS:  Dest                - Pointer to the destination DEVICE_ID
 *              Source              - Pointer to the source DEVICE_ID
 *              StringArea          - Pointer to where to copy the dest string
 *
 * RETURN:      Pointer to the next string area
 *
 * DESCRIPTION: Copy a single PNP_DEVICE_ID, including the string data.
 * DESCRIPTION: Copy a single DEVICE_ID, including the string data.
 *
 ******************************************************************************/

static char *
AcpiNsCopyDeviceId (
    ACPI_PNP_DEVICE_ID      *Dest,
    ACPI_PNP_DEVICE_ID      *Source,
    ACPI_DEVICE_ID          *Dest,
    ACPI_DEVICE_ID          *Source,
    char                    *StringArea)
{
    /* Create the destination DEVICE_ID */

    /* Create the destination PNP_DEVICE_ID */

    Dest->String = StringArea;
    Dest->Length = Source->Length;

    /* Copy actual string and return a pointer to the next string area */

    ACPI_MEMCPY (StringArea, Source->String, Source->Length);
    return (StringArea + Source->Length);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiGetObjectInfo
 *
 * PARAMETERS:  Handle              - Object Handle
 *              ReturnBuffer        - Where the info is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Returns information about an object as gleaned from the
 *              namespace node and possibly by running several standard
 *              control methods (Such as in the case of a device.)
 *
 * For Device and Processor objects, run the Device _HID, _UID, _CID, _SUB,
 * _STA, _ADR, _SxW, and _SxD methods.
 * For Device and Processor objects, run the Device _HID, _UID, _CID, _STA,
 * _ADR, _SxW, and _SxD methods.
 *
 * Note: Allocates the return buffer, must be freed by the caller.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiGetObjectInfo (
    ACPI_HANDLE             Handle,
    ACPI_DEVICE_INFO        **ReturnBuffer)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_DEVICE_INFO        *Info;
    ACPI_PNP_DEVICE_ID_LIST *CidList = NULL;
    ACPI_PNP_DEVICE_ID      *Hid = NULL;
    ACPI_PNP_DEVICE_ID      *Uid = NULL;
```



```
    ACPI_PNP_DEVICE_ID      *Sub = NULL;
    ACPI_DEVICE_ID_LIST     *CidList = NULL;
    ACPI_DEVICE_ID          *Hid = NULL;
    ACPI_DEVICE_ID          *Uid = NULL;
    char                    *NextIdString;
    ACPI_OBJECT_TYPE        Type;
    ACPI_NAME               Name;
    UINT8                   ParamCount= 0;
    UINT8                   Valid = 0;
    UINT32                  InfoSize;
    UINT32                  i;
    ACPI_STATUS             Status;


    /* Parameter validation */

    if (!Handle || !ReturnBuffer)
    {
        return (AE_BAD_PARAMETER);
    }

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
        goto Cleanup;
    }

    Node = AcpiNsValidateHandle (Handle);
    if (!Node)
    {
        (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
        return (AE_BAD_PARAMETER);
    }

    /* Get the namespace node data while the namespace is locked */

    InfoSize = sizeof (ACPI_DEVICE_INFO);
    Type = Node->Type;
    Name = Node->Name.Integer;

    if (Node->Type == ACPI_TYPE_METHOD)
    {
        ParamCount = Node->Object->Method.ParamCount;
    }

    Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    if ((Type == ACPI_TYPE_DEVICE) ||
        (Type == ACPI_TYPE_PROCESSOR))
    {
        /*
         * Get extra info for ACPI Device/Processor objects only:
         * Run the Device _HID, _UID, _SUB, and _CID methods.
         * Run the Device _HID, _UID, and _CID methods.
         *
         * Note: none of these methods are required, so they may or may
         * not be present for this device. The Info->Valid bitfield is used
         * to indicate which methods were found and run successfully.
         */

        /* Execute the Device._HID method */
```

```
372         Status = AcpiUtExecute_HID (Node, &Hid);
373         if (ACPI_SUCCESS (Status))
374         {
375             InfoSize += Hid->Length;
376             Valid |= ACPI_VALID_HID;
377         }

379         /* Execute the Device._UID method */

381         Status = AcpiUtExecute_UID (Node, &Uid);
382         if (ACPI_SUCCESS (Status))
383         {
384             InfoSize += Uid->Length;
385             Valid |= ACPI_VALID_UID;
386         }

388         /* Execute the Device._SUB method */

390         Status = AcpiUtExecute_SUB (Node, &Sub);
391         if (ACPI_SUCCESS (Status))
392         {
393             InfoSize += Sub->Length;
394             Valid |= ACPI_VALID_SUB;
395         }

397         /* Execute the Device._CID method */

399         Status = AcpiUtExecute_CID (Node, &CidList);
400         if (ACPI_SUCCESS (Status))
401         {
402             /* Add size of CID strings and CID pointer array */

404             InfoSize += (CidList->ListSize - sizeof (ACPI_PNP_DEVICE_ID_LIST));
391             InfoSize += (CidList->ListSize - sizeof (ACPI_DEVICE_ID_LIST));
405             Valid |= ACPI_VALID_CID;
406         }
407     }

409     /*
410      * Now that we have the variable-length data, we can allocate the
411      * return buffer
412      */
413     Info = ACPI_ALLOCATE_ZEROED (InfoSize);
414     if (!Info)
415     {
416         Status = AE_NO_MEMORY;
417         goto Cleanup;
418     }

420     /* Get the fixed-length data */

422     if ((Type == ACPI_TYPE_DEVICE) ||
423         (Type == ACPI_TYPE_PROCESSOR))
424     {
425         /*
426          * Get extra info for ACPI Device/Processor objects only:
427          * Run the _STA, _ADR and, SxW, and _SxD methods.
428          *
429          * Notes: none of these methods are required, so they may or may
416          * Note: none of these methods are required, so they may or may
430          * not be present for this device. The Info->Valid bitfield is used
431          * to indicate which methods were found and run successfully.
432          *
433          * For _STA, if the method does not exist, then (as per the ACPI
434          * specification), the returned CurrentStatus flags will indicate
```



```
435          * that the device is present/functional/enabled. Otherwise, the
436          * CurrentStatus flags reflect the value returned from _STA.
437          */

439         /* Execute the Device._STA method */

441         Status = AcpiUtExecute_STA (Node, &Info->CurrentStatus);
442         if (ACPI_SUCCESS (Status))
443         {
444             Valid |= ACPI_VALID_STA;
445         }

447         /* Execute the Device._ADR method */

449         Status = AcpiUtEvaluateNumericObject (METHOD_NAME__ADR, Node,
450                     &Info->Address);
451         if (ACPI_SUCCESS (Status))
452         {
453             Valid |= ACPI_VALID_ADR;
454         }

456         /* Execute the Device._SxW methods */

458         Status = AcpiUtExecutePowerMethods (Node,
459                     AcpiGbl_LowestDstateNames, ACPI_NUM_SxW_METHODS,
460                     Info->LowestDstates);
461         if (ACPI_SUCCESS (Status))
462         {
463             Valid |= ACPI_VALID_SXWS;
464         }

466         /* Execute the Device._SxD methods */

468         Status = AcpiUtExecutePowerMethods (Node,
469                     AcpiGbl_HighestDstateNames, ACPI_NUM_SxD_METHODS,
470                     Info->HighestDstates);
471         if (ACPI_SUCCESS (Status))
472         {
473             Valid |= ACPI_VALID_SXDS;
474         }
475     }

477     /*
478      * Create a pointer to the string area of the return buffer.
479      * Point to the end of the base ACPI_DEVICE_INFO structure.
480      */
481     NextIdString = ACPI_CAST_PTR (char, Info->CompatibleIdList.Ids);
482     if (CidList)
483     {
484         /* Point past the CID PNP_DEVICE_ID array */
466         /* Point past the CID DEVICE_ID array */

486         NextIdString += ((ACPI_SIZE) CidList->Count * sizeof (ACPI_PNP_DEVICE_ID
468         NextIdString += ((ACPI_SIZE) CidList->Count * sizeof (ACPI_DEVICE_ID));
487     }

489     /*
490      * Copy the HID, UID, SUB, and CIDs to the return buffer.
491      * The variable-length strings are copied to the reserved area
492      * at the end of the buffer.
472      * Copy the HID, UID, and CIDs to the return buffer. The variable-length
473      * strings are copied to the reserved area at the end of the buffer.
493      *
494      * For HID and CID, check if the ID is a PCI Root Bridge.
495      */
496     if (Hid)
```

```
497     {
498         NextIdString = AcpiNsCopyDeviceId (&Info->HardwareId,
499             Hid, NextIdString);

501         if (AcpiUtIsPciRootBridge (Hid->String))
502         {
503             Info->Flags |= ACPI_PCI_ROOT_BRIDGE;
504         }
505     }

507     if (Uid)
508     {
509         NextIdString = AcpiNsCopyDeviceId (&Info->UniqueId,
510             Uid, NextIdString);
511     }

513     if (Sub)
514     {
515         NextIdString = AcpiNsCopyDeviceId (&Info->SubsystemId,
516             Sub, NextIdString);
517     }

519     if (CidList)
520     {
521         Info->CompatibleIdList.Count = CidList->Count;
522         Info->CompatibleIdList.ListSize = CidList->ListSize;

524         /* Copy each CID */

526         for (i = 0; i < CidList->Count; i++)
527         {
528             NextIdString = AcpiNsCopyDeviceId (&Info->CompatibleIdList.Ids[i],
529                 &CidList->Ids[i], NextIdString);

531             if (AcpiUtIsPciRootBridge (CidList->Ids[i].String))
532             {
533                 Info->Flags |= ACPI_PCI_ROOT_BRIDGE;
534             }
535         }
536     }

538     /* Copy the fixed-length data */

540     Info->InfoSize = InfoSize;
541     Info->Type = Type;
542     Info->Name = Name;
543     Info->ParamCount = ParamCount;
544     Info->Valid = Valid;

546     *ReturnBuffer = Info;
547     Status = AE_OK;


550 Cleanup:
551     if (Hid)
552     {
553         ACPI_FREE (Hid);
554     }
555     if (Uid)
556     {
557         ACPI_FREE (Uid);
558     }
559     if (Sub)
560     {
561         ACPI_FREE (Sub);
562     }
```



```
563     if (CidList)
564     {
565         ACPI_FREE (CidList);
566     }
567     return (Status);
568 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    7714 Thu Dec 26 13:49:23 2013
new/usr/src/common/acpica/components/namespace/nsxfobj.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: nsxfobj - Public interfaces to the ACPI subsystem
 *                        ACPI Object oriented interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __NSXFOBJ_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_NAMESPACE
        ACPI_MODULE_NAME    ("nsxfobj")

/******************************************************************************
 *
```



```
 * FUNCTION:    AcpiGetType
 *
 * PARAMETERS:  Handle          - Handle of object whose type is desired
 *              RetType         - Where the type will be placed
 *
 * RETURN:      Status
 *
 * DESCRIPTION: This routine returns the type associatd with a particular handle
 *
 *****************************************************************************/

ACPI_STATUS
AcpiGetType (
    ACPI_HANDLE             Handle,
    ACPI_OBJECT_TYPE        *RetType)
{
    ACPI_NAMESPACE_NODE     *Node;
    ACPI_STATUS             Status;


    /* Parameter Validation */

    if (!RetType)
    {
        return (AE_BAD_PARAMETER);
    }

    /*
     * Special case for the predefined Root Node
     * (return type ANY)
     */
    if (Handle == ACPI_ROOT_OBJECT)
    {
        *RetType = ACPI_TYPE_ANY;
        return (AE_OK);
    }

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Convert and validate the handle */

    Node = AcpiNsValidateHandle (Handle);
    if (!Node)
    {
        (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
        return (AE_BAD_PARAMETER);
    }

    *RetType = Node->Type;


    Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    return (Status);
}
______unchanged_portion_omitted_

ACPI_EXPORT_SYMBOL (AcpiGetNextObject)
```

```
**********************************************************
   25895 Thu Dec 26 13:49:23 2013
new/usr/src/common/acpica/components/parser/psargs.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: psargs - Parse AML opcode arguments
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __PSARGS_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acparser.h"
  49 #include "amlcode.h"
  50 #include "acnamesp.h"
  51 #include "acdispat.h"

  53 #define _COMPONENT          ACPI_PARSER
  54         ACPI_MODULE_NAME    ("psargs")

  56 /* Local prototypes */

  58 static UINT32
  59 AcpiPsGetNextPackageLength (
```



```
  60     ACPI_PARSE_STATE        *ParserState);

  62 static ACPI_PARSE_OBJECT *
  63 AcpiPsGetNextField (
  64     ACPI_PARSE_STATE        *ParserState);


  67 /*******************************************************************************
  68  *
  69  * FUNCTION:    AcpiPsGetNextPackageLength
  70  *
  71  * PARAMETERS:  ParserState         - Current parser state object
  72  *
  73  * RETURN:      Decoded package length. On completion, the AML pointer points
  74  *              past the length byte or bytes.
  75  *
  76  * DESCRIPTION: Decode and return a package length field.
  77  *              Note: Largest package length is 28 bits, from ACPI specification
  78  *
  79  ******************************************************************************/

  81 static UINT32
  82 AcpiPsGetNextPackageLength (
  83     ACPI_PARSE_STATE        *ParserState)
  84 {
  85     UINT8                   *Aml = ParserState->Aml;
  86     UINT32                  PackageLength = 0;
  87     UINT32                  ByteCount;
  88     UINT8                   ByteZeroMask = 0x3F; /* Default [0:5] */


  91     ACPI_FUNCTION_TRACE (PsGetNextPackageLength);


  94     /*
  95      * Byte 0 bits [6:7] contain the number of additional bytes
  96      * used to encode the package length, either 0,1,2, or 3
  97      */
  98     ByteCount = (Aml[0] >> 6);
  99     ParserState->Aml += ((ACPI_SIZE) ByteCount + 1);

 101     /* Get bytes 3, 2, 1 as needed */

 103     while (ByteCount)
 104     {
 105         /*
 106          * Final bit positions for the package length bytes:
 107          *      Byte3->[20:27]
 108          *      Byte2->[12:19]
 109          *      Byte1->[04:11]
 110          *      Byte0->[00:03]
 111          */
 112         PackageLength |= (Aml[ByteCount] << ((ByteCount << 3) - 4));

 114         ByteZeroMask = 0x0F; /* Use bits [0:3] of byte 0 */
 115         ByteCount--;
 116     }

 118     /* Byte 0 is a special case, either bits [0:3] or [0:5] are used */

 120     PackageLength |= (Aml[0] & ByteZeroMask);
 121     return_UINT32 (PackageLength);
 122 }
```

*_____unchanged_portion_omitted_*

```
157 /*******************************************************************************
158  *
159  * FUNCTION:    AcpiPsGetNextNamestring
160  *
161  * PARAMETERS:  ParserState         - Current parser state object
162  *
163  * RETURN:      Pointer to the start of the name string (pointer points into
164  *              the AML.
165  *
166  * DESCRIPTION: Get next raw namestring within the AML stream. Handles all name
167  *              prefix characters. Set parser state to point past the string.
168  *              (Name is consumed from the AML.)
169  *
170  ******************************************************************************/

172 char *
173 AcpiPsGetNextNamestring (
174     ACPI_PARSE_STATE        *ParserState)
175 {
176     UINT8                   *Start = ParserState->Aml;
177     UINT8                   *End = ParserState->Aml;


180     ACPI_FUNCTION_TRACE (PsGetNextNamestring);


183     /* Point past any namestring prefix characters (backslash or carat) */

185     while (ACPI_IS_ROOT_PREFIX (*End) ||
186             ACPI_IS_PARENT_PREFIX (*End))
185     while (AcpiPsIsPrefixChar (*End))
187     {
188         End++;
189     }

191     /* Decode the path prefix character */

193     switch (*End)
194     {
195     case 0:

197         /* NullName */

199         if (End == Start)
200         {
201             Start = NULL;
202         }
203         End++;
204         break;

206     case AML_DUAL_NAME_PREFIX:

208         /* Two name segments */

210         End += 1 + (2 * ACPI_NAME_SIZE);
211         break;

213     case AML_MULTI_NAME_PREFIX_OP:

215         /* Multiple name segments, 4 chars each, count in next byte */

217         End += 2 + (*(End + 1) * ACPI_NAME_SIZE);
218         break;

220     default:
```



```
222         /* Single name segment */

224         End += ACPI_NAME_SIZE;
225         break;
226     }

228     ParserState->Aml = End;
229     return_PTR ((char *) Start);
230 }
```

*unchanged_portion_omitted_*

```
411 /*******************************************************************************
412  *
413  * FUNCTION:    AcpiPsGetNextSimpleArg
414  *
415  * PARAMETERS:  ParserState         - Current parser state object
416  *              ArgType             - The argument type (AML_*_ARG)
417  *              Arg                 - Where the argument is returned
418  *
419  * RETURN:      None
420  *
421  * DESCRIPTION: Get the next simple argument (constant, string, or namestring)
422  *
423  ******************************************************************************/

425 void
426 AcpiPsGetNextSimpleArg (
427     ACPI_PARSE_STATE        *ParserState,
428     UINT32                  ArgType,
429     ACPI_PARSE_OBJECT       *Arg)
430 {
431     UINT32                  Length;
432     UINT16                  Opcode;
433     UINT8                   *Aml = ParserState->Aml;


436     ACPI_FUNCTION_TRACE_U32 (PsGetNextSimpleArg, ArgType);


439     switch (ArgType)
440     {
441     case ARGP_BYTEDATA:

443         /* Get 1 byte from the AML stream */

445         Opcode = AML_BYTE_OP;
446         Arg->Common.Value.Integer = (UINT64) *Aml;
447         Length = 1;
448         break;


450     case ARGP_WORDDATA:

452         /* Get 2 bytes from the AML stream */

454         Opcode = AML_WORD_OP;
455         ACPI_MOVE_16_TO_64 (&Arg->Common.Value.Integer, Aml);
456         Length = 2;
457         break;


459     case ARGP_DWORDDATA:

461         /* Get 4 bytes from the AML stream */
```

```
463         Opcode = AML_DWORD_OP;
464         ACPI_MOVE_32_TO_64 (&Arg->Common.Value.Integer, Aml);
465         Length = 4;
466         break;


468     case ARGP_QWORDDATA:

470         /* Get 8 bytes from the AML stream */

472         Opcode = AML_QWORD_OP;
473         ACPI_MOVE_64_TO_64 (&Arg->Common.Value.Integer, Aml);
474         Length = 8;
475         break;


477     case ARGP_CHARLIST:

479         /* Get a pointer to the string, point past the string */

481         Opcode = AML_STRING_OP;
482         Arg->Common.Value.String = ACPI_CAST_PTR (char, Aml);

484         /* Find the null terminator */

486         Length = 0;
487         while (Aml[Length])
488         {
489             Length++;
490         }
491         Length++;
492         break;


494     case ARGP_NAME:
495     case ARGP_NAMESTRING:

497         AcpiPsInitOp (Arg, AML_INT_NAMEPATH_OP);
498         Arg->Common.Value.Name = AcpiPsGetNextNamestring (ParserState);
499         return_VOID;


501     default:

503         ACPI_ERROR ((AE_INFO, "Invalid ArgType 0x%X", ArgType));
504         return_VOID;
505     }

507     AcpiPsInitOp (Arg, Opcode);
508     ParserState->Aml += Length;
509     return_VOID;
510 }


513 /*******************************************************************************
514  *
515  * FUNCTION:    AcpiPsGetNextField
516  *
517  * PARAMETERS:  ParserState         - Current parser state object
518  *
519  * RETURN:      A newly allocated FIELD op
520  *
521  * DESCRIPTION: Get next field (NamedField, ReservedField, or AccessField)
522  *
523  ******************************************************************************/
```



```
525 static ACPI_PARSE_OBJECT *
526 AcpiPsGetNextField (
527     ACPI_PARSE_STATE        *ParserState)
528 {
529     UINT32                  AmlOffset;
534     UINT32                  AmlOffset = (UINT32)
535                                 ACPI_PTR_DIFF (ParserState->Aml,
536                                                ParserState->AmlStart);
530     ACPI_PARSE_OBJECT       *Field;
531     ACPI_PARSE_OBJECT       *Arg = NULL;
532     UINT16                  Opcode;
533     UINT32                  Name;
534     UINT8                   AccessType;
535     UINT8                   AccessAttribute;
536     UINT8                   AccessLength;
537     UINT32                  PkgLength;
538     UINT8                   *PkgEnd;
539     UINT32                  BufferLength;


542     ACPI_FUNCTION_TRACE (PsGetNextField);


545     AmlOffset = (UINT32) ACPI_PTR_DIFF (
546         ParserState->Aml, ParserState->AmlStart);

548     /* Determine field type */

550     switch (ACPI_GET8 (ParserState->Aml))
551     {
552     case AML_FIELD_OFFSET_OP:
549     default:

554         Opcode = AML_INT_RESERVEDFIELD_OP;
555         ParserState->Aml++;
551         Opcode = AML_INT_NAMEDFIELD_OP;
556         break;

558     case AML_FIELD_ACCESS_OP:
554     case 0x00:

560         Opcode = AML_INT_ACCESSFIELD_OP;
556         Opcode = AML_INT_RESERVEDFIELD_OP;
561         ParserState->Aml++;
562         break;

564     case AML_FIELD_CONNECTION_OP:
560     case 0x01:

566         Opcode = AML_INT_CONNECTION_OP;
562         Opcode = AML_INT_ACCESSFIELD_OP;
567         ParserState->Aml++;
568         break;

570     case AML_FIELD_EXT_ACCESS_OP:

572         Opcode = AML_INT_EXTACCESSFIELD_OP;
573         ParserState->Aml++;
574         break;

576     default:

578         Opcode = AML_INT_NAMEDFIELD_OP;
579         break;
580     }
```

```
582     /* Allocate a new field op */

584     Field = AcpiPsAllocOp (Opcode);
585     if (!Field)
586     {
587         return_PTR (NULL);
588     }

590     Field->Common.AmlOffset = AmlOffset;

592     /* Decode the field type */

594     switch (Opcode)
595     {
596     case AML_INT_NAMEDFIELD_OP:

598         /* Get the 4-character name */

600         ACPI_MOVE_32_TO_32 (&Name, ParserState->Aml);
601         AcpiPsSetName (Field, Name);
602         ParserState->Aml += ACPI_NAME_SIZE;

604         /* Get the length which is encoded as a package length */

606         Field->Common.Value.Size = AcpiPsGetNextPackageLength (ParserState);
607         break;


610     case AML_INT_RESERVEDFIELD_OP:

612         /* Get the length which is encoded as a package length */

614         Field->Common.Value.Size = AcpiPsGetNextPackageLength (ParserState);
615         break;


618     case AML_INT_ACCESSFIELD_OP:
619     case AML_INT_EXTACCESSFIELD_OP:

621         /*
622          * Get AccessType and AccessAttrib and merge into the field Op
623          * AccessType is first operand, AccessAttribute is second. stuff
624          * these bytes into the node integer value for convenience.
607          * AccessType is first operand, AccessAttribute is second
625          */

627         /* Get the two bytes (Type/Attribute) */

629         AccessType = ACPI_GET8 (ParserState->Aml);
609         Field->Common.Value.Integer = (((UINT32) ACPI_GET8 (ParserState->Aml) <<
630         ParserState->Aml++;
631         AccessAttribute = ACPI_GET8 (ParserState->Aml);
611         Field->Common.Value.Integer |= ACPI_GET8 (ParserState->Aml);
632         ParserState->Aml++;

634         Field->Common.Value.Integer = (UINT8) AccessType;
635         Field->Common.Value.Integer |= (UINT16) (AccessAttribute << 8);

637         /* This opcode has a third byte, AccessLength */

639         if (Opcode == AML_INT_EXTACCESSFIELD_OP)
640         {
641             AccessLength = ACPI_GET8 (ParserState->Aml);
642             ParserState->Aml++;

644             Field->Common.Value.Integer |= (UINT32) (AccessLength << 16);
```



```
645         }
646         break;


649     case AML_INT_CONNECTION_OP:

651         /*
652          * Argument for Connection operator can be either a Buffer
653          * (resource descriptor), or a NameString.
654          */
655         if (ACPI_GET8 (ParserState->Aml) == AML_BUFFER_OP)
656         {
657             ParserState->Aml++;

659             PkgEnd = ParserState->Aml;
660             PkgLength = AcpiPsGetNextPackageLength (ParserState);
661             PkgEnd += PkgLength;

663             if (ParserState->Aml < PkgEnd)
664             {
665                 /* Non-empty list */

667                 Arg = AcpiPsAllocOp (AML_INT_BYTELIST_OP);
668                 if (!Arg)
669                 {
670                     AcpiPsFreeOp (Field);
671                     return_PTR (NULL);
672                 }

674                 /* Get the actual buffer length argument */

676                 Opcode = ACPI_GET8 (ParserState->Aml);
677                 ParserState->Aml++;

679                 switch (Opcode)
680                 {
681                 case AML_BYTE_OP:       /* AML_BYTEDATA_ARG */

683                     BufferLength = ACPI_GET8 (ParserState->Aml);
684                     ParserState->Aml += 1;
685                     break;

687                 case AML_WORD_OP:       /* AML_WORDDATA_ARG */

689                     BufferLength = ACPI_GET16 (ParserState->Aml);
690                     ParserState->Aml += 2;
691                     break;

693                 case AML_DWORD_OP:      /* AML_DWORDATA_ARG */

695                     BufferLength = ACPI_GET32 (ParserState->Aml);
696                     ParserState->Aml += 4;
697                     break;

699                 default:

701                     BufferLength = 0;
702                     break;
703                 }

705                 /* Fill in bytelist data */

707                 Arg->Named.Value.Size = BufferLength;
708                 Arg->Named.Data = ParserState->Aml;
709             }
```

```
711             /* Skip to End of byte data */

713             ParserState->Aml = PkgEnd;
714         }
715         else
716         {
717             Arg = AcpiPsAllocOp (AML_INT_NAMEPATH_OP);
718             if (!Arg)
719             {
720                 AcpiPsFreeOp (Field);
721                 return_PTR (NULL);
722             }

724             /* Get the Namestring argument */

726             Arg->Common.Value.Name = AcpiPsGetNextNamestring (ParserState);
727         }

729         /* Link the buffer/namestring to parent (CONNECTION_OP) */

731         AcpiPsAppendArg (Field, Arg);
732         break;


735     default:

737         /* Opcode was set in previous switch */
738         break;
739     }

741     return_PTR (Field);
742 }


745 /*******************************************************************************
746  *
747  * FUNCTION:    AcpiPsGetNextArg
748  *
749  * PARAMETERS:  WalkState           - Current state
750  *              ParserState         - Current parser state object
751  *              ArgType             - The argument type (AML_*_ARG)
752  *              ReturnArg           - Where the next arg is returned
753  *
754  * RETURN:      Status, and an op object containing the next argument.
755  *
756  * DESCRIPTION: Get next argument (including complex list arguments that require
757  *              pushing the parser stack)
758  *
759  ******************************************************************************/

761 ACPI_STATUS
762 AcpiPsGetNextArg (
763     ACPI_WALK_STATE         *WalkState,
764     ACPI_PARSE_STATE        *ParserState,
765     UINT32                  ArgType,
766     ACPI_PARSE_OBJECT       **ReturnArg)
767 {
768     ACPI_PARSE_OBJECT       *Arg = NULL;
769     ACPI_PARSE_OBJECT       *Prev = NULL;
770     ACPI_PARSE_OBJECT       *Field;
771     UINT32                  Subop;
772     ACPI_STATUS             Status = AE_OK;


775     ACPI_FUNCTION_TRACE_PTR (PsGetNextArg, ParserState);
```



```
778     switch (ArgType)
779     {
780     case ARGP_BYTEDATA:
781     case ARGP_WORDDATA:
782     case ARGP_DWORDDATA:
783     case ARGP_CHARLIST:
784     case ARGP_NAME:
785     case ARGP_NAMESTRING:

787         /* Constants, strings, and namestrings are all the same size */

789         Arg = AcpiPsAllocOp (AML_BYTE_OP);
790         if (!Arg)
791         {
792             return_ACPI_STATUS (AE_NO_MEMORY);
793         }
794         AcpiPsGetNextSimpleArg (ParserState, ArgType, Arg);
795         break;


797     case ARGP_PKGLENGTH:

799         /* Package length, nothing returned */

801         ParserState->PkgEnd = AcpiPsGetNextPackageEnd (ParserState);
802         break;


804     case ARGP_FIELDLIST:

806         if (ParserState->Aml < ParserState->PkgEnd)
807         {
808             /* Non-empty list */

810             while (ParserState->Aml < ParserState->PkgEnd)
811             {
812                 Field = AcpiPsGetNextField (ParserState);
813                 if (!Field)
814                 {
815                     return_ACPI_STATUS (AE_NO_MEMORY);
816                 }

818                 if (Prev)
819                 {
820                     Prev->Common.Next = Field;
821                 }
822                 else
823                 {
824                     Arg = Field;
825                 }
826                 Prev = Field;
827             }

829             /* Skip to End of byte data */

831             ParserState->Aml = ParserState->PkgEnd;
832         }
833         break;


835     case ARGP_BYTELIST:

837         if (ParserState->Aml < ParserState->PkgEnd)
838         {
839             /* Non-empty list */
```

```
841             Arg = AcpiPsAllocOp (AML_INT_BYTELIST_OP);
842             if (!Arg)
843             {
844                 return_ACPI_STATUS (AE_NO_MEMORY);
845             }

847             /* Fill in bytelist data */

849             Arg->Common.Value.Size = (UINT32)
850                 ACPI_PTR_DIFF (ParserState->PkgEnd, ParserState->Aml);
851             Arg->Named.Data = ParserState->Aml;

853             /* Skip to End of byte data */

855             ParserState->Aml = ParserState->PkgEnd;
856         }
857         break;


859     case ARGP_TARGET:
860     case ARGP_SUPERNAME:
861     case ARGP_SIMPLENAME:

863         Subop = AcpiPsPeekOpcode (ParserState);
864         if (Subop == 0                  ||
865             AcpiPsIsLeadingChar (Subop) ||
866             ACPI_IS_ROOT_PREFIX (Subop) ||
867             ACPI_IS_PARENT_PREFIX (Subop))
750             AcpiPsIsPrefixChar (Subop))
868         {
869             /* NullName or NameString */

871             Arg = AcpiPsAllocOp (AML_INT_NAMEPATH_OP);
872             if (!Arg)
873             {
874                 return_ACPI_STATUS (AE_NO_MEMORY);
875             }

877             /* To support SuperName arg of Unload */

879             if (WalkState->Opcode == AML_UNLOAD_OP)
880             {
881                 Status = AcpiPsGetNextNamepath (WalkState, ParserState, Arg, 1);

883                 /*
884                  * If the SuperName arg of Unload is a method call,
885                  * we have restored the AML pointer, just free this Arg
886                  */
887                 if (Arg->Common.AmlOpcode == AML_INT_METHODCALL_OP)
888                 {
889                     AcpiPsFreeOp (Arg);
890                     Arg = NULL;
891                 }
892             }
893             else
894             {
895                 Status = AcpiPsGetNextNamepath (WalkState, ParserState, Arg, 0);
896             }
897         }
898         else
899         {
900             /* Single complex argument, nothing returned */

902             WalkState->ArgCount = 1;
903         }
```



```
904         break;


906     case ARGP_DATAOBJ:
907     case ARGP_TERMARG:

909         /* Single complex argument, nothing returned */

911         WalkState->ArgCount = 1;
912         break;


914     case ARGP_DATAOBJLIST:
915     case ARGP_TERMLIST:
916     case ARGP_OBJLIST:

918         if (ParserState->Aml < ParserState->PkgEnd)
919         {
920             /* Non-empty list of variable arguments, nothing returned */

922             WalkState->ArgCount = ACPI_VAR_ARGS;
923         }
924         break;


926     default:

928         ACPI_ERROR ((AE_INFO, "Invalid ArgType: 0x%X", ArgType));
929         Status = AE_AML_OPERAND_TYPE;
930         break;
931     }

933     *ReturnArg = Arg;
934     return_ACPI_STATUS (Status);
935 }
```

unchanged_portion_omitted_

```
**********************************************************
   20478 Thu Dec 26 13:49:24 2013
new/usr/src/common/acpica/components/parser/psloop.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: psloop - Main AML parse loop
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


/*
 * Parse the AML and build an operation tree as most interpreters, (such as
 * Perl) do. Parsing is done by hand rather than with a YACC generated parser
 * to tightly constrain stack and dynamic memory usage. Parsing is kept
 * flexible and the code fairly compact by parsing based on a list of AML
 * opcode templates in AmlOpInfo[].
 */

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acdispat.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psloop")
```



```
/* Local prototypes */

static ACPI_STATUS
AcpiPsGetArguments (
    ACPI_WALK_STATE         *WalkState,
    UINT8                   *AmlOpStart,
    ACPI_PARSE_OBJECT       *Op);

static void
AcpiPsLinkModuleCode (
    ACPI_PARSE_OBJECT       *ParentOp,
    UINT8                   *AmlStart,
    UINT32                  AmlLength,
    ACPI_OWNER_ID           OwnerId);


/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetArguments
 *
 * PARAMETERS:  WalkState           - Current state
 *              AmlOpStart          - Op start in AML
 *              Op                  - Current Op
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Get arguments for passed Op.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiPsGetArguments (
    ACPI_WALK_STATE         *WalkState,
    UINT8                   *AmlOpStart,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_PARSE_OBJECT       *Arg = NULL;
    const ACPI_OPCODE_INFO  *OpInfo;


    ACPI_FUNCTION_TRACE_PTR (PsGetArguments, WalkState);


    switch (Op->Common.AmlOpcode)
    {
    case AML_BYTE_OP:       /* AML_BYTEDATA_ARG */
    case AML_WORD_OP:       /* AML_WORDDATA_ARG */
    case AML_DWORD_OP:      /* AML_DWORDATA_ARG */
    case AML_QWORD_OP:      /* AML_QWORDATA_ARG */
    case AML_STRING_OP:     /* AML_ASCIICHARLIST_ARG */

        /* Fill in constant or string argument directly */

        AcpiPsGetNextSimpleArg (&(WalkState->ParserState),
            GET_CURRENT_ARG_TYPE (WalkState->ArgTypes), Op);
        break;

    case AML_INT_NAMEPATH_OP:   /* AML_NAMESTRING_ARG */

        Status = AcpiPsGetNextNamepath (WalkState, &(WalkState->ParserState), Op
        if (ACPI_FAILURE (Status))
        {
            return_ACPI_STATUS (Status);
```

```
127         }
129         WalkState->ArgTypes = 0;
130         break;
132     default:
133         /*
134          * Op is not a constant or string, append each argument to the Op
135          */
136         while (GET_CURRENT_ARG_TYPE (WalkState->ArgTypes) && !WalkState->ArgCoun
137         {
138             WalkState->AmlOffset = (UINT32) ACPI_PTR_DIFF (WalkState->ParserStat
139                 WalkState->ParserState.AmlStart);
141             Status = AcpiPsGetNextArg (WalkState, &(WalkState->ParserState),
142                         GET_CURRENT_ARG_TYPE (WalkState->ArgTypes), &Arg);
143             if (ACPI_FAILURE (Status))
144             {
145                 return_ACPI_STATUS (Status);
146             }
148             if (Arg)
149             {
150                 Arg->Common.AmlOffset = WalkState->AmlOffset;
151                 AcpiPsAppendArg (Op, Arg);
152             }
154             INCREMENT_ARG_LIST (WalkState->ArgTypes);
155         }
158         /*
159          * Handle executable code at "module-level". This refers to
160          * executable opcodes that appear outside of any control method.
161          */
162         if ((WalkState->PassNumber <= ACPI_IMODE_LOAD_PASS2) &&
163             ((WalkState->ParseFlags & ACPI_PARSE_DISASSEMBLE) == 0))
164         {
165             /*
166              * We want to skip If/Else/While constructs during Pass1 because we
167              * want to actually conditionally execute the code during Pass2.
168              *
169              * Except for disassembly, where we always want to walk the
170              * If/Else/While packages
171              */
172             switch (Op->Common.AmlOpcode)
173             {
174             case AML_IF_OP:
175             case AML_ELSE_OP:
176             case AML_WHILE_OP:
177                 /*
178                  * Currently supported module-level opcodes are:
179                  * IF/ELSE/WHILE. These appear to be the most common,
180                  * and easiest to support since they open an AML
181                  * package.
182                  */
183                 if (WalkState->PassNumber == ACPI_IMODE_LOAD_PASS1)
184                 {
185                     AcpiPsLinkModuleCode (Op->Common.Parent, AmlOpStart,
186                         (UINT32) (WalkState->ParserState.PkgEnd - AmlOpStart),
187                         WalkState->OwnerId);
188                 }
190                 ACPI_DEBUG_PRINT ((ACPI_DB_PARSE,
191                     "Pass1: Skipping an If/Else/While body\n"));
```



```
193                 /* Skip body of if/else/while in pass 1 */
195                 WalkState->ParserState.Aml = WalkState->ParserState.PkgEnd;
196                 WalkState->ArgCount = 0;
197                 break;
199             default:
200                 /*
201                  * Check for an unsupported executable opcode at module
202                  * level. We must be in PASS1, the parent must be a SCOPE,
203                  * The opcode class must be EXECUTE, and the opcode must
204                  * not be an argument to another opcode.
205                  */
206                 if ((WalkState->PassNumber == ACPI_IMODE_LOAD_PASS1) &&
207                     (Op->Common.Parent->Common.AmlOpcode == AML_SCOPE_OP))
208                 {
209                     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
210                     if ((OpInfo->Class == AML_CLASS_EXECUTE) &&
211                         (!Arg))
212                     {
213                         ACPI_WARNING ((AE_INFO,
214                             "Unsupported module-level executable opcode "
215                             "0x%.2X at table offset 0x%.4X",
216                             Op->Common.AmlOpcode,
217                             (UINT32) (ACPI_PTR_DIFF (AmlOpStart,
218                                 WalkState->ParserState.AmlStart) +
219                                 sizeof (ACPI_TABLE_HEADER))));
220                     }
221                 }
222                 break;
223             }
224         }
226         /* Special processing for certain opcodes */
228         switch (Op->Common.AmlOpcode)
229         {
230         case AML_METHOD_OP:
231             /*
232              * Skip parsing of control method because we don’t have enough
233              * info in the first pass to parse it correctly.
234              *
235              * Save the length and address of the body
236              */
237             Op->Named.Data = WalkState->ParserState.Aml;
238             Op->Named.Length = (UINT32)
239                 (WalkState->ParserState.PkgEnd - WalkState->ParserState.Aml);
241             /* Skip body of method */
243             WalkState->ParserState.Aml = WalkState->ParserState.PkgEnd;
244             WalkState->ArgCount = 0;
245             break;
247         case AML_BUFFER_OP:
248         case AML_PACKAGE_OP:
249         case AML_VAR_PACKAGE_OP:
251             if ((Op->Common.Parent) &&
252                 (Op->Common.Parent->Common.AmlOpcode == AML_NAME_OP) &&
253                 (WalkState->PassNumber <= ACPI_IMODE_LOAD_PASS2))
254             {
255                 /*
256                  * Skip parsing of Buffers and Packages because we don’t have
257                  * enough info in the first pass to parse them correctly.
258                  */
```

```
259                 Op->Named.Data = AmlOpStart;
260                 Op->Named.Length = (UINT32)
261                     (WalkState->ParserState.PkgEnd - AmlOpStart);

263                 /* Skip body */

265                 WalkState->ParserState.Aml = WalkState->ParserState.PkgEnd;
266                 WalkState->ArgCount = 0;
267             }
268             break;

270         case AML_WHILE_OP:

272             if (WalkState->ControlState)
273             {
274                 WalkState->ControlState->Control.PackageEnd =
275                     WalkState->ParserState.PkgEnd;
276             }
277             break;

279         default:

281             /* No action for all other opcodes */

283             break;
284         }

286         break;
287     }

289     return_ACPI_STATUS (AE_OK);
290 }


293 /*******************************************************************************
294  *
295  * FUNCTION:    AcpiPsLinkModuleCode
296  *
297  * PARAMETERS:  ParentOp            - Parent parser op
298  *              AmlStart            - Pointer to the AML
299  *              AmlLength           - Length of executable AML
300  *              OwnerId             - OwnerId of module level code
301  *
302  * RETURN:      None.
303  *
304  * DESCRIPTION: Wrap the module-level code with a method object and link the
305  *              object to the global list. Note, the mutex field of the method
306  *              object is used to link multiple module-level code objects.
307  *
308  ******************************************************************************/

310 static void
311 AcpiPsLinkModuleCode (
312     ACPI_PARSE_OBJECT       *ParentOp,
313     UINT8                   *AmlStart,
314     UINT32                  AmlLength,
315     ACPI_OWNER_ID           OwnerId)
316 {
317     ACPI_OPERAND_OBJECT     *Prev;
318     ACPI_OPERAND_OBJECT     *Next;
319     ACPI_OPERAND_OBJECT     *MethodObj;
320     ACPI_NAMESPACE_NODE     *ParentNode;


323     /* Get the tail of the list */
```



```
325     Prev = Next = AcpiGbl_ModuleCodeList;
326     while (Next)
327     {
328         Prev = Next;
329         Next = Next->Method.Mutex;
330     }

332     /*
333      * Insert the module level code into the list. Merge it if it is
334      * adjacent to the previous element.
335      */
336     if (!Prev ||
337        ((Prev->Method.AmlStart + Prev->Method.AmlLength) != AmlStart))
338     {
339         /* Create, initialize, and link a new temporary method object */

341         MethodObj = AcpiUtCreateInternalObject (ACPI_TYPE_METHOD);
342         if (!MethodObj)
343         {
344             return;
345         }

347         if (ParentOp->Common.Node)
348         {
349             ParentNode = ParentOp->Common.Node;
350         }
351         else
352         {
353             ParentNode = AcpiGbl_RootNode;
354         }

356         MethodObj->Method.AmlStart = AmlStart;
357         MethodObj->Method.AmlLength = AmlLength;
358         MethodObj->Method.OwnerId = OwnerId;
359         MethodObj->Method.InfoFlags |= ACPI_METHOD_MODULE_LEVEL;

361         /*
362          * Save the parent node in NextObject. This is cheating, but we
363          * don't want to expand the method object.
364          */
365         MethodObj->Method.NextObject =
366             ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, ParentNode);

368         if (!Prev)
369         {
370             AcpiGbl_ModuleCodeList = MethodObj;
371         }
372         else
373         {
374             Prev->Method.Mutex = MethodObj;
375         }
376     }
377     else
378     {
379         Prev->Method.AmlLength += AmlLength;
380     }
381 }

383 /*******************************************************************************
384  *
385  * FUNCTION:    AcpiPsParseLoop
386  *
387  * PARAMETERS:  WalkState           - Current state
388  *
389  * RETURN:      Status
390  *
```

```
 * DESCRIPTION: Parse AML (pointed to by the current parser state) and return
 *              a tree of ops.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiPsParseLoop (
    ACPI_WALK_STATE         *WalkState)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_PARSE_OBJECT       *Op = NULL;     /* current op */
    ACPI_PARSE_STATE        *ParserState;
    UINT8                   *AmlOpStart = NULL;


    ACPI_FUNCTION_TRACE_PTR (PsParseLoop, WalkState);


    if (WalkState->DescendingCallback == NULL)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    ParserState = &WalkState->ParserState;
    WalkState->ArgTypes = 0;

#if (!defined (ACPI_NO_METHOD_EXECUTION) && !defined (ACPI_CONSTANT_EVAL_ONLY))

    if (WalkState->WalkType & ACPI_WALK_METHOD_RESTART)
    {
        /* We are restarting a preempted control method */

        if (AcpiPsHasCompletedScope (ParserState))
        {
            /*
             * We must check if a predicate to an IF or WHILE statement
             * was just completed
             */
            if ((ParserState->Scope->ParseScope.Op) &&
               ((ParserState->Scope->ParseScope.Op->Common.AmlOpcode == AML_IF_O
                (ParserState->Scope->ParseScope.Op->Common.AmlOpcode == AML_WHIL
                (WalkState->ControlState) &&
                (WalkState->ControlState->Common.State ==
                    ACPI_CONTROL_PREDICATE_EXECUTING))
            {
                /*
                 * A predicate was just completed, get the value of the
                 * predicate and branch based on that value
                 */
                WalkState->Op = NULL;
                Status = AcpiDsGetPredicateValue (WalkState, ACPI_TO_POINTER (TR
                if (ACPI_FAILURE (Status) &&
                    ((Status & AE_CODE_MASK) != AE_CODE_CONTROL))
                {
                    if (Status == AE_AML_NO_RETURN_VALUE)
                    {
                        ACPI_EXCEPTION ((AE_INFO, Status,
                            "Invoked method did not return a value"));
                    }

                    ACPI_EXCEPTION ((AE_INFO, Status, "GetPredicate Failed"));
                    return_ACPI_STATUS (Status);
                }

                Status = AcpiPsNextParseState (WalkState, Op, Status);
            }
```



```
            AcpiPsPopScope (ParserState, &Op,
                &WalkState->ArgTypes, &WalkState->ArgCount);
            ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "Popped scope, Op=%p\n", Op));
        }
        else if (WalkState->PrevOp)
        {
            /* We were in the middle of an op */

            Op = WalkState->PrevOp;
            WalkState->ArgTypes = WalkState->PrevArgTypes;
        }
    }
#endif

    /* Iterative parsing loop, while there is more AML to process: */

    while ((ParserState->Aml < ParserState->AmlEnd) || (Op))
    {
        AmlOpStart = ParserState->Aml;
        if (!Op)
        {
            Status = AcpiPsCreateOp (WalkState, AmlOpStart, &Op);
            if (ACPI_FAILURE (Status))
            {
                if (Status == AE_CTRL_PARSE_CONTINUE)
                {
                    continue;
                }

                if (Status == AE_CTRL_PARSE_PENDING)
                {
                    Status = AE_OK;
                }

                Status = AcpiPsCompleteOp (WalkState, &Op, Status);
                if (ACPI_FAILURE (Status))
                {
                    return_ACPI_STATUS (Status);
                }

                continue;
            }

            Op->Common.AmlOffset = WalkState->AmlOffset;

            if (WalkState->OpInfo)
            {
                ACPI_DEBUG_PRINT ((ACPI_DB_PARSE,
                    "Opcode %4.4X [%s] Op %p Aml %p AmlOffset %5.5X\n",
                     (UINT32) Op->Common.AmlOpcode, WalkState->OpInfo->Name,
                     Op, ParserState->Aml, Op->Common.AmlOffset));
            }
        }


        /*
         * Start ArgCount at zero because we don't know if there are
         * any args yet
         */
        WalkState->ArgCount  = 0;

        /* Are there any arguments that must be processed? */

        if (WalkState->ArgTypes)
        {
```

```
523             /* Get arguments */
524
525             Status = AcpiPsGetArguments (WalkState, AmlOpStart, Op);
526             if (ACPI_FAILURE (Status))
527             {
528                 Status = AcpiPsCompleteOp (WalkState, &Op, Status);
529                 if (ACPI_FAILURE (Status))
530                 {
531                     return_ACPI_STATUS (Status);
532                 }
533
534                 continue;
535             }
536         }
537
538         /* Check for arguments that need to be processed */
539
540         if (WalkState->ArgCount)
541         {
542             /*
543              * There are arguments (complex ones), push Op and
544              * prepare for argument
545              */
546             Status = AcpiPsPushScope (ParserState, Op,
547                         WalkState->ArgTypes, WalkState->ArgCount);
548             if (ACPI_FAILURE (Status))
549             {
550                 Status = AcpiPsCompleteOp (WalkState, &Op, Status);
551                 if (ACPI_FAILURE (Status))
552                 {
553                     return_ACPI_STATUS (Status);
554                 }
555
556                 continue;
557             }
558
559             Op = NULL;
560             continue;
561         }
562
563         /*
564          * All arguments have been processed -- Op is complete,
565          * prepare for next
566          */
567         WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
568         if (WalkState->OpInfo->Flags & AML_NAMED)
569         {
570             if (Op->Common.AmlOpcode == AML_REGION_OP ||
571                 Op->Common.AmlOpcode == AML_DATA_REGION_OP)
572             {
573                 /*
574                  * Skip parsing of control method or opregion body,
575                  * because we don't have enough info in the first pass
576                  * to parse them correctly.
577                  *
578                  * Completed parsing an OpRegion declaration, we now
579                  * know the length.
580                  */
581                 Op->Named.Length = (UINT32) (ParserState->Aml - Op->Named.Data);
582             }
583         }
584
585         if (WalkState->OpInfo->Flags & AML_CREATE)
586         {
587             /*
588              * Backup to beginning of CreateXXXfield declaration (1 for
```



```
589              * Opcode)
590              *
591              * BodyLength is unknown until we parse the body
592              */
593             Op->Named.Length = (UINT32) (ParserState->Aml - Op->Named.Data);
594         }
595
596         if (Op->Common.AmlOpcode == AML_BANK_FIELD_OP)
597         {
598             /*
599              * Backup to beginning of BankField declaration
600              *
601              * BodyLength is unknown until we parse the body
602              */
603             Op->Named.Length = (UINT32) (ParserState->Aml - Op->Named.Data);
604         }
605
606         /* This op complete, notify the dispatcher */
607
608         if (WalkState->AscendingCallback != NULL)
609         {
610             WalkState->Op = Op;
611             WalkState->Opcode = Op->Common.AmlOpcode;
612
613             Status = WalkState->AscendingCallback (WalkState);
614             Status = AcpiPsNextParseState (WalkState, Op, Status);
615             if (Status == AE_CTRL_PENDING)
616             {
617                 Status = AE_OK;
618             }
619         }
620
621         Status = AcpiPsCompleteOp (WalkState, &Op, Status);
622         if (ACPI_FAILURE (Status))
623         {
624             return_ACPI_STATUS (Status);
625         }
626
627     } /* while ParserState->Aml */
628
629     Status = AcpiPsCompleteFinalOp (WalkState, Op, Status);
630     return_ACPI_STATUS (Status);
631 }
```

```
**********************************************************
   19963 Thu Dec 26 13:49:24 2013
new/usr/src/common/acpica/components/parser/psobject.c
acpica-unix2-20130823
**********************************************************
/******************************************************************************
 *
 * Module Name: psobject - Support for parse objects
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psobject")


/* Local prototypes */

static ACPI_STATUS
AcpiPsGetAmlOpcode (
    ACPI_WALK_STATE         *WalkState);


/******************************************************************************
```



```
 *
 * FUNCTION:    AcpiPsGetAmlOpcode
 *
 * PARAMETERS:  WalkState           - Current state
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Extract the next AML opcode from the input stream.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiPsGetAmlOpcode (
    ACPI_WALK_STATE         *WalkState)
{

    ACPI_FUNCTION_TRACE_PTR (PsGetAmlOpcode, WalkState);


    WalkState->AmlOffset = (UINT32) ACPI_PTR_DIFF (WalkState->ParserState.Aml,
                                                   WalkState->ParserState.AmlStart);
    WalkState->Opcode = AcpiPsPeekOpcode (&(WalkState->ParserState));

    /*
     * First cut to determine what we have found:
     * 1) A valid AML opcode
     * 2) A name string
     * 3) An unknown/invalid opcode
     */
    WalkState->OpInfo = AcpiPsGetOpcodeInfo (WalkState->Opcode);

    switch (WalkState->OpInfo->Class)
    {
    case AML_CLASS_ASCII:
    case AML_CLASS_PREFIX:
        /*
         * Starts with a valid prefix or ASCII char, this is a name
         * string. Convert the bare name string to a namepath.
         */
        WalkState->Opcode = AML_INT_NAMEPATH_OP;
        WalkState->ArgTypes = ARGP_NAMESTRING;
        break;

    case AML_CLASS_UNKNOWN:

        /* The opcode is unrecognized. Complain and skip unknown opcodes */

        if (WalkState->PassNumber == 2)
        {
            ACPI_ERROR ((AE_INFO,
                "Unknown opcode 0x%.2X at table offset 0x%.4X, ignoring",
                WalkState->Opcode,
                (UINT32) (WalkState->AmlOffset + sizeof (ACPI_TABLE_HEADER))));

            ACPI_DUMP_BUFFER ((WalkState->ParserState.Aml - 16), 48);

#ifdef ACPI_ASL_COMPILER
            /*
             * This is executed for the disassembler only. Output goes
             * to the disassembled ASL output file.
             */
            AcpiOsPrintf (
                "/*\nError: Unknown opcode 0x%.2X at table offset 0x%.4X, contex
                WalkState->Opcode,
                (UINT32) (WalkState->AmlOffset + sizeof (ACPI_TABLE_HEADER)));
```

```
128             /* Dump the context surrounding the invalid opcode */

130             AcpiUtDumpBuffer (((UINT8 *) WalkState->ParserState.Aml - 16),
131                 48, DB_BYTE_DISPLAY,
132                 (WalkState->AmlOffset + sizeof (ACPI_TABLE_HEADER) - 16));
133             AcpiOsPrintf (" */\n");
134 #endif
135         }

137         /* Increment past one-byte or two-byte opcode */

139         WalkState->ParserState.Aml++;
140         if (WalkState->Opcode > 0xFF) /* Can only happen if first byte is 0x5B *
141         {
142             WalkState->ParserState.Aml++;
143         }

145         return_ACPI_STATUS (AE_CTRL_PARSE_CONTINUE);

147     default:

149         /* Found opcode info, this is a normal opcode */

151         WalkState->ParserState.Aml += AcpiPsGetOpcodeSize (WalkState->Opcode);
152         WalkState->ArgTypes = WalkState->OpInfo->ParseArgs;
153         break;
154     }

156     return_ACPI_STATUS (AE_OK);
157 }


160 /*******************************************************************************
161  *
162  * FUNCTION:    AcpiPsBuildNamedOp
163  *
164  * PARAMETERS:  WalkState           - Current state
165  *              AmlOpStart          - Begin of named Op in AML
166  *              UnnamedOp           - Early Op (not a named Op)
167  *              Op                  - Returned Op
168  *
169  * RETURN:      Status
170  *
171  * DESCRIPTION: Parse a named Op
172  *
173  ******************************************************************************/

175 ACPI_STATUS
176 AcpiPsBuildNamedOp (
177     ACPI_WALK_STATE         *WalkState,
178     UINT8                   *AmlOpStart,
179     ACPI_PARSE_OBJECT       *UnnamedOp,
180     ACPI_PARSE_OBJECT       **Op)
181 {
182     ACPI_STATUS             Status = AE_OK;
183     ACPI_PARSE_OBJECT       *Arg = NULL;


186     ACPI_FUNCTION_TRACE_PTR (PsBuildNamedOp, WalkState);


189     UnnamedOp->Common.Value.Arg = NULL;
190     UnnamedOp->Common.ArgListLength = 0;
191     UnnamedOp->Common.AmlOpcode = WalkState->Opcode;

193     /*
```



```
194      * Get and append arguments until we find the node that contains
195      * the name (the type ARGP_NAME).
196      */
197     while (GET_CURRENT_ARG_TYPE (WalkState->ArgTypes) &&
198            (GET_CURRENT_ARG_TYPE (WalkState->ArgTypes) != ARGP_NAME))
199     {
200         Status = AcpiPsGetNextArg (WalkState, &(WalkState->ParserState),
201                     GET_CURRENT_ARG_TYPE (WalkState->ArgTypes), &Arg);
202         if (ACPI_FAILURE (Status))
203         {
204             return_ACPI_STATUS (Status);
205         }

207         AcpiPsAppendArg (UnnamedOp, Arg);
208         INCREMENT_ARG_LIST (WalkState->ArgTypes);
209     }

211     /*
212      * Make sure that we found a NAME and didn't run out of arguments
213      */
214     if (!GET_CURRENT_ARG_TYPE (WalkState->ArgTypes))
215     {
216         return_ACPI_STATUS (AE_AML_NO_OPERAND);
217     }

219     /* We know that this arg is a name, move to next arg */

221     INCREMENT_ARG_LIST (WalkState->ArgTypes);

223     /*
224      * Find the object. This will either insert the object into
225      * the namespace or simply look it up
226      */
227     WalkState->Op = NULL;

229     Status = WalkState->DescendingCallback (WalkState, Op);
230     if (ACPI_FAILURE (Status))
231     {
232         ACPI_EXCEPTION ((AE_INFO, Status, "During name lookup/catalog"));
233         return_ACPI_STATUS (Status);
234     }

236     if (!*Op)
237     {
238         return_ACPI_STATUS (AE_CTRL_PARSE_CONTINUE);
239     }

241     Status = AcpiPsNextParseState (WalkState, *Op, Status);
242     if (ACPI_FAILURE (Status))
243     {
244         if (Status == AE_CTRL_PENDING)
245         {
246             return_ACPI_STATUS (AE_CTRL_PARSE_PENDING);
247         }
248         return_ACPI_STATUS (Status);
249     }

251     AcpiPsAppendArg (*Op, UnnamedOp->Common.Value.Arg);

253     if ((*Op)->Common.AmlOpcode == AML_REGION_OP ||
254         (*Op)->Common.AmlOpcode == AML_DATA_REGION_OP)
255     {
256         /*
257          * Defer final parsing of an OperationRegion body, because we don't
258          * have enough info in the first pass to parse it correctly (i.e.,
259          * there may be method calls within the TermArg elements of the body.)
```

```
260          *
261          * However, we must continue parsing because the opregion is not a
262          * standalone package -- we don't know where the end is at this point.
263          *
264          * (Length is unknown until parse of the body complete)
265          */
266         (*Op)->Named.Data = AmlOpStart;
267         (*Op)->Named.Length = 0;
268     }

270     return_ACPI_STATUS (AE_OK);
271 }


274 /*****************************************************************************
275  *
276  * FUNCTION:    AcpiPsCreateOp
277  *
278  * PARAMETERS:  WalkState           - Current state
279  *              AmlOpStart          - Op start in AML
280  *              NewOp               - Returned Op
281  *
282  * RETURN:      Status
283  *
284  * DESCRIPTION: Get Op from AML
285  *
286  ****************************************************************************/

288 ACPI_STATUS
289 AcpiPsCreateOp (
290     ACPI_WALK_STATE         *WalkState,
291     UINT8                   *AmlOpStart,
292     ACPI_PARSE_OBJECT       **NewOp)
293 {
294     ACPI_STATUS             Status = AE_OK;
295     ACPI_PARSE_OBJECT       *Op;
296     ACPI_PARSE_OBJECT       *NamedOp = NULL;
297     ACPI_PARSE_OBJECT       *ParentScope;
298     UINT8                   ArgumentCount;
299     const ACPI_OPCODE_INFO  *OpInfo;


302     ACPI_FUNCTION_TRACE_PTR (PsCreateOp, WalkState);


305     Status = AcpiPsGetAmlOpcode (WalkState);
306     if (Status == AE_CTRL_PARSE_CONTINUE)
307     {
308         return_ACPI_STATUS (AE_CTRL_PARSE_CONTINUE);
309     }

311     /* Create Op structure and append to parent's argument list */

313     WalkState->OpInfo = AcpiPsGetOpcodeInfo (WalkState->Opcode);
314     Op = AcpiPsAllocOp (WalkState->Opcode);
315     if (!Op)
316     {
317         return_ACPI_STATUS (AE_NO_MEMORY);
318     }

320     if (WalkState->OpInfo->Flags & AML_NAMED)
321     {
322         Status = AcpiPsBuildNamedOp (WalkState, AmlOpStart, Op, &NamedOp);
323         AcpiPsFreeOp (Op);
324         if (ACPI_FAILURE (Status))
325         {
```



```
326             return_ACPI_STATUS (Status);
327         }

329         *NewOp = NamedOp;
330         return_ACPI_STATUS (AE_OK);
331     }

333     /* Not a named opcode, just allocate Op and append to parent */

335     if (WalkState->OpInfo->Flags & AML_CREATE)
336     {
337         /*
338          * Backup to beginning of CreateXXXfield declaration
339          * BodyLength is unknown until we parse the body
340          */
341         Op->Named.Data = AmlOpStart;
342         Op->Named.Length = 0;
343     }

345     if (WalkState->Opcode == AML_BANK_FIELD_OP)
346     {
347         /*
348          * Backup to beginning of BankField declaration
349          * BodyLength is unknown until we parse the body
350          */
351         Op->Named.Data = AmlOpStart;
352         Op->Named.Length = 0;
353     }

355     ParentScope = AcpiPsGetParentScope (&(WalkState->ParserState));
356     AcpiPsAppendArg (ParentScope, Op);

358     if (ParentScope)
359     {
360         OpInfo = AcpiPsGetOpcodeInfo (ParentScope->Common.AmlOpcode);
361         if (OpInfo->Flags & AML_HAS_TARGET)
362         {
363             ArgumentCount = AcpiPsGetArgumentCount (OpInfo->Type);
364             if (ParentScope->Common.ArgListLength > ArgumentCount)
365             {
366                 Op->Common.Flags |= ACPI_PARSEOP_TARGET;
367             }
368         }
369         else if (ParentScope->Common.AmlOpcode == AML_INCREMENT_OP)
370         {
371             Op->Common.Flags |= ACPI_PARSEOP_TARGET;
372         }
373     }

375     if (WalkState->DescendingCallback != NULL)
376     {
377         /*
378          * Find the object. This will either insert the object into
379          * the namespace or simply look it up
380          */
381         WalkState->Op = *NewOp = Op;

383         Status = WalkState->DescendingCallback (WalkState, &Op);
384         Status = AcpiPsNextParseState (WalkState, Op, Status);
385         if (Status == AE_CTRL_PENDING)
386         {
387             Status = AE_CTRL_PARSE_PENDING;
388         }
389     }

391     return_ACPI_STATUS (Status);
```

```
392 }


395 /*******************************************************************************
396  *
397  * FUNCTION:    AcpiPsCompleteOp
398  *
399  * PARAMETERS:  WalkState           - Current state
400  *              Op                  - Returned Op
401  *              Status              - Parse status before complete Op
402  *
403  * RETURN:      Status
404  *
405  * DESCRIPTION: Complete Op
406  *
407  ******************************************************************************/

409 ACPI_STATUS
410 AcpiPsCompleteOp (
411     ACPI_WALK_STATE         *WalkState,
412     ACPI_PARSE_OBJECT       **Op,
413     ACPI_STATUS             Status)
414 {
415     ACPI_STATUS             Status2;


418     ACPI_FUNCTION_TRACE_PTR (PsCompleteOp, WalkState);


421     /*
422      * Finished one argument of the containing scope
423      */
424     WalkState->ParserState.Scope->ParseScope.ArgCount--;

426     /* Close this Op (will result in parse subtree deletion) */

428     Status2 = AcpiPsCompleteThisOp (WalkState, *Op);
429     if (ACPI_FAILURE (Status2))
430     {
431         return_ACPI_STATUS (Status2);
432     }

434     *Op = NULL;

436     switch (Status)
437     {
438     case AE_OK:

440         break;

442     case AE_CTRL_TRANSFER:

444         /* We are about to transfer to a called method */

446         WalkState->PrevOp = NULL;
447         WalkState->PrevArgTypes = WalkState->ArgTypes;
448         return_ACPI_STATUS (Status);

450     case AE_CTRL_END:

452         AcpiPsPopScope (&(WalkState->ParserState), Op,
453             &WalkState->ArgTypes, &WalkState->ArgCount);

455         if (*Op)
456         {
457             WalkState->Op = *Op;
```



```
458             WalkState->OpInfo = AcpiPsGetOpcodeInfo ((*Op)->Common.AmlOpcode);
459             WalkState->Opcode = (*Op)->Common.AmlOpcode;

461             Status = WalkState->AscendingCallback (WalkState);
462             Status = AcpiPsNextParseState (WalkState, *Op, Status);

464             Status2 = AcpiPsCompleteThisOp (WalkState, *Op);
465             if (ACPI_FAILURE (Status2))
466             {
467                 return_ACPI_STATUS (Status2);
468             }
469         }

471         Status = AE_OK;
472         break;

474     case AE_CTRL_BREAK:
475     case AE_CTRL_CONTINUE:

477         /* Pop off scopes until we find the While */

479         while (!(*Op) || ((*Op)->Common.AmlOpcode != AML_WHILE_OP))
480         {
481             AcpiPsPopScope (&(WalkState->ParserState), Op,
482                 &WalkState->ArgTypes, &WalkState->ArgCount);
483         }

485         /* Close this iteration of the While loop */

487         WalkState->Op = *Op;
488         WalkState->OpInfo = AcpiPsGetOpcodeInfo ((*Op)->Common.AmlOpcode);
489         WalkState->Opcode = (*Op)->Common.AmlOpcode;

491         Status = WalkState->AscendingCallback (WalkState);
492         Status = AcpiPsNextParseState (WalkState, *Op, Status);

494         Status2 = AcpiPsCompleteThisOp (WalkState, *Op);
495         if (ACPI_FAILURE (Status2))
496         {
497             return_ACPI_STATUS (Status2);
498         }

500         Status = AE_OK;
501         break;

503     case AE_CTRL_TERMINATE:

505         /* Clean up */
506         do
507         {
508             if (*Op)
509             {
510                 Status2 = AcpiPsCompleteThisOp (WalkState, *Op);
511                 if (ACPI_FAILURE (Status2))
512                 {
513                     return_ACPI_STATUS (Status2);
514                 }

516                 AcpiUtDeleteGenericState (
517                     AcpiUtPopGenericState (&WalkState->ControlState));
518             }

520             AcpiPsPopScope (&(WalkState->ParserState), Op,
521                 &WalkState->ArgTypes, &WalkState->ArgCount);

523         } while (*Op);
```

```
525         return_ACPI_STATUS (AE_OK);

527     default:  /* All other non-AE_OK status */

529         do
530         {
531             if (*Op)
532             {
533                 Status2 = AcpiPsCompleteThisOp (WalkState, *Op);
534                 if (ACPI_FAILURE (Status2))
535                 {
536                     return_ACPI_STATUS (Status2);
537                 }
538             }

540             AcpiPsPopScope (&(WalkState->ParserState), Op,
541                 &WalkState->ArgTypes, &WalkState->ArgCount);

543         } while (*Op);


546 #if 0
547         /*
548          * TBD: Cleanup parse ops on error
549          */
550         if (*Op == NULL)
551         {
552             AcpiPsPopScope (ParserState, Op,
553                 &WalkState->ArgTypes, &WalkState->ArgCount);
554         }
555 #endif
556         WalkState->PrevOp = NULL;
557         WalkState->PrevArgTypes = WalkState->ArgTypes;
558         return_ACPI_STATUS (Status);
559     }

561     /* This scope complete? */

563     if (AcpiPsHasCompletedScope (&(WalkState->ParserState)))
564     {
565         AcpiPsPopScope (&(WalkState->ParserState), Op,
566             &WalkState->ArgTypes, &WalkState->ArgCount);
567         ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "Popped scope, Op=%p\n", *Op));
568     }
569     else
570     {
571         *Op = NULL;
572     }

574     return_ACPI_STATUS (AE_OK);
575 }


578 /*******************************************************************************
579  *
580  * FUNCTION:    AcpiPsCompleteFinalOp
581  *
582  * PARAMETERS:  WalkState           - Current state
583  *              Op                  - Current Op
584  *              Status              - Current parse status before complete last
585  *                                    Op
586  *
587  * RETURN:      Status
588  *
589  * DESCRIPTION: Complete last Op.
```



```
590  *
591  ******************************************************************************/

593 ACPI_STATUS
594 AcpiPsCompleteFinalOp (
595     ACPI_WALK_STATE         *WalkState,
596     ACPI_PARSE_OBJECT       *Op,
597     ACPI_STATUS             Status)
598 {
599     ACPI_STATUS             Status2;


602     ACPI_FUNCTION_TRACE_PTR (PsCompleteFinalOp, WalkState);


605     /*
606      * Complete the last Op (if not completed), and clear the scope stack.
607      * It is easily possible to end an AML "package" with an unbounded number
608      * of open scopes (such as when several ASL blocks are closed with
609      * sequential closing braces). We want to terminate each one cleanly.
610      */
611     ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "AML package complete at Op %p\n", Op));
612     do
613     {
614         if (Op)
615         {
616             if (WalkState->AscendingCallback != NULL)
617             {
618                 WalkState->Op = Op;
619                 WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
620                 WalkState->Opcode = Op->Common.AmlOpcode;

622                 Status = WalkState->AscendingCallback (WalkState);
623                 Status = AcpiPsNextParseState (WalkState, Op, Status);
624                 if (Status == AE_CTRL_PENDING)
625                 {
626                     Status = AcpiPsCompleteOp (WalkState, &Op, AE_OK);
627                     if (ACPI_FAILURE (Status))
628                     {
629                         return_ACPI_STATUS (Status);
630                     }
631                 }

633                 if (Status == AE_CTRL_TERMINATE)
634                 {
635                     Status = AE_OK;

637                     /* Clean up */
638                     do
639                     {
640                         if (Op)
641                         {
642                             Status2 = AcpiPsCompleteThisOp (WalkState, Op);
643                             if (ACPI_FAILURE (Status2))
644                             {
645                                 return_ACPI_STATUS (Status2);
646                             }
647                         }

649                         AcpiPsPopScope (&(WalkState->ParserState), &Op,
650                             &WalkState->ArgTypes, &WalkState->ArgCount);

652                     } while (Op);

654                     return_ACPI_STATUS (Status);
655                 }
```

```
657                 else if (ACPI_FAILURE (Status))
658                 {
659                     /* First error is most important */
661                     (void) AcpiPsCompleteThisOp (WalkState, Op);
662                     return_ACPI_STATUS (Status);
663                 }
664             }
666             Status2 = AcpiPsCompleteThisOp (WalkState, Op);
667             if (ACPI_FAILURE (Status2))
668             {
669                 return_ACPI_STATUS (Status2);
670             }
671         }
673         AcpiPsPopScope (&(WalkState->ParserState), &Op, &WalkState->ArgTypes,
674             &WalkState->ArgCount);
676     } while (Op);
678     return_ACPI_STATUS (Status);
679 }
```

****
33177 Thu Dec 26 13:49:24 2013
new/usr/src/common/acpica/components/parser/psopcode.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: psopcode - Parser/Interpreter opcode information table
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #include "acpi.h"
  46 #include "accommon.h"
  47 #include "acparser.h"
  47 #include "acopcode.h"
  48 #include "amlcode.h"


  51 #define _COMPONENT          ACPI_PARSER
  52         ACPI_MODULE_NAME    ("psopcode")


  56 static const UINT8          AcpiGbl_ArgumentCount[] = {0,1,1,1,1,2,2,2,2,3,3,6};


  55 /******************************************************************************
```



```
  56  *
  57  * NAME:        AcpiGbl_AmlOpInfo
  58  *
  59  * DESCRIPTION: Opcode table. Each entry contains <opcode, type, name, operands>
  60  *              The name is a simple ascii string, the operand specifier is an
  61  *              ascii string with one letter per operand. The letter specifies
  62  *              the operand type.
  63  *
  64  *****************************************************************************/

  66 /*
  67  * Summary of opcode types/flags
  68  *

  70  Opcodes that have associated namespace objects (AML_NSOBJECT flag)

  72     AML_SCOPE_OP
  73     AML_DEVICE_OP
  74     AML_THERMAL_ZONE_OP
  75     AML_METHOD_OP
  76     AML_POWER_RES_OP
  77     AML_PROCESSOR_OP
  78     AML_FIELD_OP
  79     AML_INDEX_FIELD_OP
  80     AML_BANK_FIELD_OP
  81     AML_NAME_OP
  82     AML_ALIAS_OP
  83     AML_MUTEX_OP
  84     AML_EVENT_OP
  85     AML_REGION_OP
  86     AML_CREATE_FIELD_OP
  87     AML_CREATE_BIT_FIELD_OP
  88     AML_CREATE_BYTE_FIELD_OP
  89     AML_CREATE_WORD_FIELD_OP
  90     AML_CREATE_DWORD_FIELD_OP
  91     AML_CREATE_QWORD_FIELD_OP
  92     AML_INT_NAMEDFIELD_OP
  93     AML_INT_METHODCALL_OP
  94     AML_INT_NAMEPATH_OP

  96   Opcodes that are "namespace" opcodes (AML_NSOPCODE flag)

  98     AML_SCOPE_OP
  99     AML_DEVICE_OP
 100     AML_THERMAL_ZONE_OP
 101     AML_METHOD_OP
 102     AML_POWER_RES_OP
 103     AML_PROCESSOR_OP
 104     AML_FIELD_OP
 105     AML_INDEX_FIELD_OP
 106     AML_BANK_FIELD_OP
 107     AML_NAME_OP
 108     AML_ALIAS_OP
 109     AML_MUTEX_OP
 110     AML_EVENT_OP
 111     AML_REGION_OP
 112     AML_INT_NAMEDFIELD_OP

 114   Opcodes that have an associated namespace node (AML_NSNODE flag)

 116     AML_SCOPE_OP
 117     AML_DEVICE_OP
 118     AML_THERMAL_ZONE_OP
 119     AML_METHOD_OP
 120     AML_POWER_RES_OP
 121     AML_PROCESSOR_OP
```

```
    AML_NAME_OP
    AML_ALIAS_OP
    AML_MUTEX_OP
    AML_EVENT_OP
    AML_REGION_OP
    AML_CREATE_FIELD_OP
    AML_CREATE_BIT_FIELD_OP
    AML_CREATE_BYTE_FIELD_OP
    AML_CREATE_WORD_FIELD_OP
    AML_CREATE_DWORD_FIELD_OP
    AML_CREATE_QWORD_FIELD_OP
    AML_INT_NAMEDFIELD_OP
    AML_INT_METHODCALL_OP
    AML_INT_NAMEPATH_OP

  Opcodes that define named ACPI objects (AML_NAMED flag)

    AML_SCOPE_OP
    AML_DEVICE_OP
    AML_THERMAL_ZONE_OP
    AML_METHOD_OP
    AML_POWER_RES_OP
    AML_PROCESSOR_OP
    AML_NAME_OP
    AML_ALIAS_OP
    AML_MUTEX_OP
    AML_EVENT_OP
    AML_REGION_OP
    AML_INT_NAMEDFIELD_OP

  Opcodes that contain executable AML as part of the definition that
  must be deferred until needed

    AML_METHOD_OP
    AML_VAR_PACKAGE_OP
    AML_CREATE_FIELD_OP
    AML_CREATE_BIT_FIELD_OP
    AML_CREATE_BYTE_FIELD_OP
    AML_CREATE_WORD_FIELD_OP
    AML_CREATE_DWORD_FIELD_OP
    AML_CREATE_QWORD_FIELD_OP
    AML_REGION_OP
    AML_BUFFER_OP

  Field opcodes

    AML_CREATE_FIELD_OP
    AML_FIELD_OP
    AML_INDEX_FIELD_OP
    AML_BANK_FIELD_OP

  Field "Create" opcodes

    AML_CREATE_FIELD_OP
    AML_CREATE_BIT_FIELD_OP
    AML_CREATE_BYTE_FIELD_OP
    AML_CREATE_WORD_FIELD_OP
    AML_CREATE_DWORD_FIELD_OP
    AML_CREATE_QWORD_FIELD_OP

 *****************************************************************************/


/*
 * Master Opcode information table. A summary of everything we know about each
 * opcode, all in one place.
```



```
 */
const ACPI_OPCODE_INFO    AcpiGbl_AmlOpInfo[AML_NUM_OPCODES] =
{
/*! [Begin] no source code translation */
/* Index           Name                  Parser Args                Interpreter Ar

/* 00 */ ACPI_OP ("Zero",               ARGP_ZERO_OP,              ARGI_ZERO_OP,
/* 01 */ ACPI_OP ("One",                ARGP_ONE_OP,               ARGI_ONE_OP,
/* 02 */ ACPI_OP ("Alias",              ARGP_ALIAS_OP,             ARGI_ALIAS_OP
/* 03 */ ACPI_OP ("Name",               ARGP_NAME_OP,              ARGI_NAME_OP,
/* 04 */ ACPI_OP ("ByteConst",          ARGP_BYTE_OP,              ARGI_BYTE_OP,
/* 05 */ ACPI_OP ("WordConst",          ARGP_WORD_OP,              ARGI_WORD_OP,
/* 06 */ ACPI_OP ("DwordConst",         ARGP_DWORD_OP,             ARGI_DWORD_OP
/* 07 */ ACPI_OP ("String",             ARGP_STRING_OP,            ARGI_STRING_O
/* 08 */ ACPI_OP ("Scope",              ARGP_SCOPE_OP,             ARGI_SCOPE_OP
/* 09 */ ACPI_OP ("Buffer",             ARGP_BUFFER_OP,            ARGI_BUFFER_O
/* 0A */ ACPI_OP ("Package",            ARGP_PACKAGE_OP,           ARGI_PACKAGE_
/* 0B */ ACPI_OP ("Method",             ARGP_METHOD_OP,            ARGI_METHOD_O
/* 0C */ ACPI_OP ("Local0",             ARGP_LOCAL0,               ARGI_LOCAL0,
/* 0D */ ACPI_OP ("Local1",             ARGP_LOCAL1,               ARGI_LOCAL1,
/* 0E */ ACPI_OP ("Local2",             ARGP_LOCAL2,               ARGI_LOCAL2,
/* 0F */ ACPI_OP ("Local3",             ARGP_LOCAL3,               ARGI_LOCAL3,
/* 10 */ ACPI_OP ("Local4",             ARGP_LOCAL4,               ARGI_LOCAL4,
/* 11 */ ACPI_OP ("Local5",             ARGP_LOCAL5,               ARGI_LOCAL5,
/* 12 */ ACPI_OP ("Local6",             ARGP_LOCAL6,               ARGI_LOCAL6,
/* 13 */ ACPI_OP ("Local7",             ARGP_LOCAL7,               ARGI_LOCAL7,
/* 14 */ ACPI_OP ("Arg0",               ARGP_ARG0,                 ARGI_ARG0,
/* 15 */ ACPI_OP ("Arg1",               ARGP_ARG1,                 ARGI_ARG1,
/* 16 */ ACPI_OP ("Arg2",               ARGP_ARG2,                 ARGI_ARG2,
/* 17 */ ACPI_OP ("Arg3",               ARGP_ARG3,                 ARGI_ARG3,
/* 18 */ ACPI_OP ("Arg4",               ARGP_ARG4,                 ARGI_ARG4,
/* 19 */ ACPI_OP ("Arg5",               ARGP_ARG5,                 ARGI_ARG5,
/* 1A */ ACPI_OP ("Arg6",               ARGP_ARG6,                 ARGI_ARG6,
/* 1B */ ACPI_OP ("Store",              ARGP_STORE_OP,             ARGI_STORE_OP
/* 1C */ ACPI_OP ("RefOf",              ARGP_REF_OF_OP,            ARGI_REF_OF_O
/* 1D */ ACPI_OP ("Add",                ARGP_ADD_OP,               ARGI_ADD_OP,
/* 1E */ ACPI_OP ("Concatenate",        ARGP_CONCAT_OP,            ARGI_CONCAT_O
/* 1F */ ACPI_OP ("Subtract",           ARGP_SUBTRACT_OP,          ARGI_SUBTRACT
/* 20 */ ACPI_OP ("Increment",          ARGP_INCREMENT_OP,         ARGI_INCREMEN
/* 21 */ ACPI_OP ("Decrement",          ARGP_DECREMENT_OP,         ARGI_DECREMEN
/* 22 */ ACPI_OP ("Multiply",           ARGP_MULTIPLY_OP,          ARGI_MULTIPLY
/* 23 */ ACPI_OP ("Divide",             ARGP_DIVIDE_OP,            ARGI_DIVIDE_O
/* 24 */ ACPI_OP ("ShiftLeft",          ARGP_SHIFT_LEFT_OP,        ARGI_SHIFT_LE
/* 25 */ ACPI_OP ("ShiftRight",         ARGP_SHIFT_RIGHT_OP,       ARGI_SHIFT_RI
/* 26 */ ACPI_OP ("And",                ARGP_BIT_AND_OP,           ARGI_BIT_AND_
/* 27 */ ACPI_OP ("NAnd",               ARGP_BIT_NAND_OP,          ARGI_BIT_NAND
/* 28 */ ACPI_OP ("Or",                 ARGP_BIT_OR_OP,            ARGI_BIT_OR_O
/* 29 */ ACPI_OP ("NOr",                ARGP_BIT_NOR_OP,           ARGI_BIT_NOR_
/* 2A */ ACPI_OP ("XOr",                ARGP_BIT_XOR_OP,           ARGI_BIT_XOR_
/* 2B */ ACPI_OP ("Not",                ARGP_BIT_NOT_OP,           ARGI_BIT_NOT_
/* 2C */ ACPI_OP ("FindSetLeftBit",     ARGP_FIND_SET_LEFT_BIT_OP, ARGI_FIND_SET
/* 2D */ ACPI_OP ("FindSetRightBit",    ARGP_FIND_SET_RIGHT_BIT_OP,ARGI_FIND_SET
/* 2E */ ACPI_OP ("DerefOf",            ARGP_DEREF_OF_OP,          ARGI_DEREF_OF
/* 2F */ ACPI_OP ("Notify",             ARGP_NOTIFY_OP,            ARGI_NOTIFY_O
/* 30 */ ACPI_OP ("SizeOf",             ARGP_SIZE_OF_OP,           ARGI_SIZE_OF_
/* 31 */ ACPI_OP ("Index",              ARGP_INDEX_OP,             ARGI_INDEX_OP
/* 32 */ ACPI_OP ("Match",              ARGP_MATCH_OP,             ARGI_MATCH_OP
/* 33 */ ACPI_OP ("CreateDWordField",   ARGP_CREATE_DWORD_FIELD_OP,ARGI_CREATE_D
/* 34 */ ACPI_OP ("CreateWordField",    ARGP_CREATE_WORD_FIELD_OP, ARGI_CREATE_W
/* 35 */ ACPI_OP ("CreateByteField",    ARGP_CREATE_BYTE_FIELD_OP, ARGI_CREATE_B
/* 36 */ ACPI_OP ("CreateBitField",     ARGP_CREATE_BIT_FIELD_OP,  ARGI_CREATE_B
/* 37 */ ACPI_OP ("ObjectType",         ARGP_TYPE_OP,              ARGI_TYPE_OP,
/* 38 */ ACPI_OP ("LAnd",               ARGP_LAND_OP,              ARGI_LAND_OP,
/* 39 */ ACPI_OP ("LOr",                ARGP_LOR_OP,               ARGI_LOR_OP,
/* 3A */ ACPI_OP ("LNot",               ARGP_LNOT_OP,              ARGI_LNOT_OP,
/* 3B */ ACPI_OP ("LEqual",             ARGP_LEQUAL_OP,            ARGI_LEQUAL_O
```

```
 254 /* 3C */ ACPI_OP ("LGreater",           ARGP_LGREATER_OP,          ARGI_LGREATER
 255 /* 3D */ ACPI_OP ("LLess",              ARGP_LLESS_OP,             ARGI_LLESS_OP
 256 /* 3E */ ACPI_OP ("If",                 ARGP_IF_OP,                ARGI_IF_OP,
 257 /* 3F */ ACPI_OP ("Else",               ARGP_ELSE_OP,              ARGI_ELSE_OP,
 258 /* 40 */ ACPI_OP ("While",              ARGP_WHILE_OP,             ARGI_WHILE_OP
 259 /* 41 */ ACPI_OP ("Noop",               ARGP_NOOP_OP,              ARGI_NOOP_OP,
 260 /* 42 */ ACPI_OP ("Return",             ARGP_RETURN_OP,            ARGI_RETURN_O
 261 /* 43 */ ACPI_OP ("Break",              ARGP_BREAK_OP,             ARGI_BREAK_OP
 262 /* 44 */ ACPI_OP ("BreakPoint",         ARGP_BREAK_POINT_OP,       ARGI_BREAK_PO
 263 /* 45 */ ACPI_OP ("Ones",               ARGP_ONES_OP,              ARGI_ONES_OP,

 265 /* Prefixed opcodes (Two-byte opcodes with a prefix op) */

 267 /* 46 */ ACPI_OP ("Mutex",              ARGP_MUTEX_OP,             ARGI_MUTEX_OP
 268 /* 47 */ ACPI_OP ("Event",              ARGP_EVENT_OP,             ARGI_EVENT_OP
 269 /* 48 */ ACPI_OP ("CondRefOf",          ARGP_COND_REF_OF_OP,       ARGI_COND_REF
 270 /* 49 */ ACPI_OP ("CreateField",        ARGP_CREATE_FIELD_OP,      ARGI_CREATE_F
 271 /* 4A */ ACPI_OP ("Load",               ARGP_LOAD_OP,              ARGI_LOAD_OP,
 272 /* 4B */ ACPI_OP ("Stall",              ARGP_STALL_OP,             ARGI_STALL_OP
 273 /* 4C */ ACPI_OP ("Sleep",              ARGP_SLEEP_OP,             ARGI_SLEEP_OP
 274 /* 4D */ ACPI_OP ("Acquire",            ARGP_ACQUIRE_OP,           ARGI_ACQUIRE_
 275 /* 4E */ ACPI_OP ("Signal",             ARGP_SIGNAL_OP,            ARGI_SIGNAL_O
 276 /* 4F */ ACPI_OP ("Wait",               ARGP_WAIT_OP,              ARGI_WAIT_OP,
 277 /* 50 */ ACPI_OP ("Reset",              ARGP_RESET_OP,             ARGI_RESET_OP
 278 /* 51 */ ACPI_OP ("Release",            ARGP_RELEASE_OP,           ARGI_RELEASE_
 279 /* 52 */ ACPI_OP ("FromBCD",            ARGP_FROM_BCD_OP,          ARGI_FROM_BCD
 280 /* 53 */ ACPI_OP ("ToBCD",              ARGP_TO_BCD_OP,            ARGI_TO_BCD_O
 281 /* 54 */ ACPI_OP ("Unload",             ARGP_UNLOAD_OP,            ARGI_UNLOAD_O
 282 /* 55 */ ACPI_OP ("Revision",           ARGP_REVISION_OP,          ARGI_REVISION
 283 /* 56 */ ACPI_OP ("Debug",              ARGP_DEBUG_OP,             ARGI_DEBUG_OP
 284 /* 57 */ ACPI_OP ("Fatal",              ARGP_FATAL_OP,             ARGI_FATAL_OP
 285 /* 58 */ ACPI_OP ("OperationRegion",    ARGP_REGION_OP,            ARGI_REGION_O
 286 /* 59 */ ACPI_OP ("Field",              ARGP_FIELD_OP,             ARGI_FIELD_OP
 287 /* 5A */ ACPI_OP ("Device",             ARGP_DEVICE_OP,            ARGI_DEVICE_O
 288 /* 5B */ ACPI_OP ("Processor",          ARGP_PROCESSOR_OP,         ARGI_PROCESSO
 289 /* 5C */ ACPI_OP ("PowerResource",      ARGP_POWER_RES_OP,         ARGI_POWER_RE
 290 /* 5D */ ACPI_OP ("ThermalZone",        ARGP_THERMAL_ZONE_OP,      ARGI_THERMAL_
 291 /* 5E */ ACPI_OP ("IndexField",         ARGP_INDEX_FIELD_OP,       ARGI_INDEX_FI
 292 /* 5F */ ACPI_OP ("BankField",          ARGP_BANK_FIELD_OP,        ARGI_BANK_FIE

 294 /* Internal opcodes that map to invalid AML opcodes */

 296 /* 60 */ ACPI_OP ("LNotEqual",          ARGP_LNOTEQUAL_OP,         ARGI_LNOTEQUA
 297 /* 61 */ ACPI_OP ("LLessEqual",         ARGP_LLESSEQUAL_OP,        ARGI_LLESSEQU
 298 /* 62 */ ACPI_OP ("LGreaterEqual",      ARGP_LGREATEREQUAL_OP,     ARGI_LGREATER
 299 /* 63 */ ACPI_OP ("-NamePath-",         ARGP_NAMEPATH_OP,          ARGI_NAMEPATH
 300 /* 64 */ ACPI_OP ("-MethodCall-",       ARGP_METHODCALL_OP,        ARGI_METHODCA
 301 /* 65 */ ACPI_OP ("-ByteList-",         ARGP_BYTELIST_OP,          ARGI_BYTELIST
 302 /* 66 */ ACPI_OP ("-ReservedField-",    ARGP_RESERVEDFIELD_OP,     ARGI_RESERVED
 303 /* 67 */ ACPI_OP ("-NamedField-",       ARGP_NAMEDFIELD_OP,        ARGI_NAMEDFIE
 304 /* 68 */ ACPI_OP ("-AccessField-",      ARGP_ACCESSFIELD_OP,       ARGI_ACCESSFI
 305 /* 69 */ ACPI_OP ("-StaticString",      ARGP_STATICSTRING_OP,      ARGI_STATICST
 306 /* 6A */ ACPI_OP ("-Return Value-",     ARG_NONE,                  ARG_NONE,
 307 /* 6B */ ACPI_OP ("-UNKNOWN_OP-",       ARG_NONE,                  ARG_NONE,
 308 /* 6C */ ACPI_OP ("-ASCII_ONLY-",       ARG_NONE,                  ARG_NONE,
 309 /* 6D */ ACPI_OP ("-PREFIX_ONLY-",      ARG_NONE,                  ARG_NONE,

 311 /* ACPI 2.0 opcodes */

 313 /* 6E */ ACPI_OP ("QwordConst",         ARGP_QWORD_OP,             ARGI_QWORD_OP
 314 /* 6F */ ACPI_OP ("Package", /* Var */  ARGP_VAR_PACKAGE_OP,       ARGI_VAR_PACK
 315 /* 70 */ ACPI_OP ("ConcatenateResTemplate", ARGP_CONCAT_RES_OP,    ARGI_CONCAT_R
 316 /* 71 */ ACPI_OP ("Mod",                ARGP_MOD_OP,               ARGI_MOD_OP,
 317 /* 72 */ ACPI_OP ("CreateQWordField",   ARGP_CREATE_QWORD_FIELD_OP,ARGI_CREATE_Q
 318 /* 73 */ ACPI_OP ("ToBuffer",           ARGP_TO_BUFFER_OP,         ARGI_TO_BUFFE
 319 /* 74 */ ACPI_OP ("ToDecimalString",    ARGP_TO_DEC_STR_OP,        ARGI_TO_DEC_S
```



```
 320 /* 75 */ ACPI_OP ("ToHexString",        ARGP_TO_HEX_STR_OP,        ARGI_TO_HEX_S
 321 /* 76 */ ACPI_OP ("ToInteger",          ARGP_TO_INTEGER_OP,        ARGI_TO_INTEG
 322 /* 77 */ ACPI_OP ("ToString",           ARGP_TO_STRING_OP,         ARGI_TO_STRIN
 323 /* 78 */ ACPI_OP ("CopyObject",         ARGP_COPY_OP,              ARGI_COPY_OP,
 324 /* 79 */ ACPI_OP ("Mid",                ARGP_MID_OP,               ARGI_MID_OP,
 325 /* 7A */ ACPI_OP ("Continue",           ARGP_CONTINUE_OP,          ARGI_CONTINUE
 326 /* 7B */ ACPI_OP ("LoadTable",          ARGP_LOAD_TABLE_OP,        ARGI_LOAD_TAB
 327 /* 7C */ ACPI_OP ("DataTableRegion",    ARGP_DATA_REGION_OP,       ARGI_DATA_REG
 328 /* 7D */ ACPI_OP ("[EvalSubTree]",      ARGP_SCOPE_OP,             ARGI_SCOPE_OP

 330 /* ACPI 3.0 opcodes */

 332 /* 7E */ ACPI_OP ("Timer",              ARGP_TIMER_OP,             ARGI_TIMER_OP
 336 /* 7E */ ACPI_OP ("Timer",              ARGP_TIMER_OP,             ARGI_TIMER_OP

 334 /* ACPI 5.0 opcodes */
 338 /*! [End] no source code translation !*/
 339 };

 336 /* 7F */ ACPI_OP ("-ConnectField-",     ARGP_CONNECTFIELD_OP,      ARGI_CONNECTF
 337 /* 80 */ ACPI_OP ("-ExtAccessField-",   ARGP_CONNECTFIELD_OP,      ARGI_CONNECTF
 341 /*
 342  * This table is directly indexed by the opcodes, and returns an
 343  * index into the table above
 344  */
 345 static const UINT8 AcpiGbl_ShortOpIndex[256] =
 346 {
 347 /*              0     1     2     3     4     5     6     7  */
 348 /*              8     9     A     B     C     D     E     F  */
 349 /* 0x00 */    0x00, 0x01, _UNK, _UNK, _UNK, _UNK, 0x02, _UNK,
 350 /* 0x08 */    0x03, _UNK, 0x04, 0x05, 0x06, 0x07, 0x6E, _UNK,
 351 /* 0x10 */    0x08, 0x09, 0x0a, 0x6F, 0x0b, _UNK, _UNK, _UNK,
 352 /* 0x18 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 353 /* 0x20 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 354 /* 0x28 */    _UNK, _UNK, _UNK, _UNK, _UNK, 0x63, _PFX, _PFX,
 355 /* 0x30 */    0x67, 0x66, 0x68, 0x65, 0x69, 0x64, 0x6A, 0x7D,
 356 /* 0x38 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 357 /* 0x40 */    _UNK, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
 358 /* 0x48 */    _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
 359 /* 0x50 */    _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
 360 /* 0x58 */    _ASC, _ASC, _ASC, _UNK, _PFX, _UNK, _PFX, _ASC,
 361 /* 0x60 */    0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13,
 362 /* 0x68 */    0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, _UNK,
 363 /* 0x70 */    0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x20, 0x21, 0x22,
 364 /* 0x78 */    0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2a,
 365 /* 0x80 */    0x2b, 0x2c, 0x2d, 0x2e, 0x70, 0x71, 0x2f, 0x30,
 366 /* 0x88 */    0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x72,
 367 /* 0x90 */    0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x73, 0x74,
 368 /* 0x98 */    0x75, 0x76, _UNK, _UNK, 0x77, 0x78, 0x79, 0x7A,
 369 /* 0xA0 */    0x3e, 0x3f, 0x40, 0x41, 0x42, 0x43, 0x60, 0x61,
 370 /* 0xA8 */    0x62, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 371 /* 0xB0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 372 /* 0xB8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 373 /* 0xC0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 374 /* 0xC8 */    _UNK, _UNK, _UNK, _UNK, 0x44, _UNK, _UNK, _UNK,
 375 /* 0xD0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 376 /* 0xD8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 377 /* 0xE0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 378 /* 0xE8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 379 /* 0xF0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
 380 /* 0xF8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x45,
 381 };

 339 /*! [End] no source code translation !*/
 383 /*
 384  * This table is indexed by the second opcode of the extended opcode
```

```
 * pair.  It returns an index into the opcode table (AcpiGbl_AmlOpInfo)
 */
static const UINT8 AcpiGbl_LongOpIndex[NUM_EXTENDED_OPCODE] =
{
/*              0     1     2     3     4     5     6     7  */
/*              8     9     A     B     C     D     E     F  */
/* 0x00 */    _UNK, 0x46, 0x47, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x08 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x10 */    _UNK, _UNK, 0x48, 0x49, _UNK, _UNK, _UNK, _UNK,
/* 0x18 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x7B,
/* 0x20 */    0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51,
/* 0x28 */    0x52, 0x53, 0x54, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x30 */    0x55, 0x56, 0x57, 0x7e, _UNK, _UNK, _UNK, _UNK,
/* 0x38 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x40 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x48 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x50 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x58 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x60 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x68 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x70 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x78 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x80 */    0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
/* 0x88 */    0x7C,
};


/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetOpcodeInfo
 *
 * PARAMETERS:  Opcode              - The AML opcode
 *
 * RETURN:      A pointer to the info about the opcode.
 *
 * DESCRIPTION: Find AML opcode description based on the opcode.
 *              NOTE: This procedure must ALWAYS return a valid pointer!
 *
 ******************************************************************************/

const ACPI_OPCODE_INFO *
AcpiPsGetOpcodeInfo (
    UINT16                  Opcode)
{
    ACPI_FUNCTION_NAME (PsGetOpcodeInfo);


    /*
     * Detect normal 8-bit opcode or extended 16-bit opcode
     */
    if (!(Opcode & 0xFF00))
    {
        /* Simple (8-bit) opcode: 0-255, can't index beyond table  */

        return (&AcpiGbl_AmlOpInfo [AcpiGbl_ShortOpIndex [(UINT8) Opcode]]);
    }

    if (((Opcode & 0xFF00) == AML_EXTENDED_OPCODE) &&
        (((UINT8) Opcode) <= MAX_EXTENDED_OPCODE))
    {
        /* Valid extended (16-bit) opcode */

        return (&AcpiGbl_AmlOpInfo [AcpiGbl_LongOpIndex [(UINT8) Opcode]]);
    }

    /* Unknown AML opcode */
```



```
    ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
        "Unknown AML opcode [%4.4X]\n", Opcode));

    return (&AcpiGbl_AmlOpInfo [_UNK]);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetOpcodeName
 *
 * PARAMETERS:  Opcode              - The AML opcode
 *
 * RETURN:      A pointer to the name of the opcode (ASCII String)
 *              Note: Never returns NULL.
 *
 * DESCRIPTION: Translate an opcode into a human-readable string
 *
 ******************************************************************************/

char *
AcpiPsGetOpcodeName (
    UINT16                  Opcode)
{
#if defined(ACPI_DISASSEMBLER) || defined (ACPI_DEBUG_OUTPUT)

    const ACPI_OPCODE_INFO  *Op;


    Op = AcpiPsGetOpcodeInfo (Opcode);

    /* Always guaranteed to return a valid pointer */

    return (Op->Name);

#else
    return ("OpcodeName unavailable");

#endif
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetArgumentCount
 *
 * PARAMETERS:  OpType              - Type associated with the AML opcode
 *
 * RETURN:      Argument count
 *
 * DESCRIPTION: Obtain the number of expected arguments for an AML opcode
 *
 ******************************************************************************/

UINT8
AcpiPsGetArgumentCount (
    UINT32                  OpType)
{

    if (OpType <= AML_TYPE_EXEC_6A_0T_1R)
    {
        return (AcpiGbl_ArgumentCount[OpType]);
    }

    return (0);
```

```
517 }
```

```
**********************************************************
    9834 Thu Dec 26 13:49:25 2013
new/usr/src/common/acpica/components/parser/psopinfo.c
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: psopinfo - AML opcode information functions and dispatch tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acopcode.h"
#include "amlcode.h"


#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psopinfo")


extern const UINT8      AcpiGbl_ShortOpIndex[];
extern const UINT8      AcpiGbl_LongOpIndex[];

static const UINT8      AcpiGbl_ArgumentCount[] = {0,1,1,1,1,2,2,2,2,3,3,6};
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiPsGetOpcodeInfo
 *
 * PARAMETERS:  Opcode              - The AML opcode
 *
 * RETURN:      A pointer to the info about the opcode.
 *
 * DESCRIPTION: Find AML opcode description based on the opcode.
 *              NOTE: This procedure must ALWAYS return a valid pointer!
 *
 *****************************************************************************/

const ACPI_OPCODE_INFO *
AcpiPsGetOpcodeInfo (
    UINT16                  Opcode)
{
#ifdef ACPI_DEBUG_OUTPUT
    const char              *OpcodeName = "Unknown AML opcode";
#endif

    ACPI_FUNCTION_NAME (PsGetOpcodeInfo);


    /*
     * Detect normal 8-bit opcode or extended 16-bit opcode
     */
    if (!(Opcode & 0xFF00))
    {
        /* Simple (8-bit) opcode: 0-255, can't index beyond table  */

        return (&AcpiGbl_AmlOpInfo [AcpiGbl_ShortOpIndex [(UINT8) Opcode]]);
    }

    if (((Opcode & 0xFF00) == AML_EXTENDED_OPCODE) &&
        (((UINT8) Opcode) <= MAX_EXTENDED_OPCODE))
    {
        /* Valid extended (16-bit) opcode */

        return (&AcpiGbl_AmlOpInfo [AcpiGbl_LongOpIndex [(UINT8) Opcode]]);
    }

#if defined ACPI_ASL_COMPILER && defined ACPI_DEBUG_OUTPUT
#include "asldefine.h"

    switch (Opcode)
    {
    case AML_RAW_DATA_BYTE:
        OpcodeName = "-Raw Data Byte-";
        break;

    case AML_RAW_DATA_WORD:
        OpcodeName = "-Raw Data Word-";
        break;

    case AML_RAW_DATA_DWORD:
        OpcodeName = "-Raw Data Dword-";
        break;

    case AML_RAW_DATA_QWORD:
        OpcodeName = "-Raw Data Qword-";
        break;

    case AML_RAW_DATA_BUFFER:
        OpcodeName = "-Raw Data Buffer-";
```

```
127         break;

129     case AML_RAW_DATA_CHAIN:
130         OpcodeName = "-Raw Data Buffer Chain-";
131         break;

133     case AML_PACKAGE_LENGTH:
134         OpcodeName = "-Package Length-";
135         break;

137     case AML_UNASSIGNED_OPCODE:
138         OpcodeName = "-Unassigned Opcode-";
139         break;

141     case AML_DEFAULT_ARG_OP:
142         OpcodeName = "-Default Arg-";
143         break;

145     default:
146         break;
147     }
148 #endif

150     /* Unknown AML opcode */

152     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
153         "%s [%4.4X]\n", OpcodeName, Opcode));

155     return (&AcpiGbl_AmlOpInfo [_UNK]);
156 }


159 /******************************************************************************
160  *
161  * FUNCTION:    AcpiPsGetOpcodeName
162  *
163  * PARAMETERS:  Opcode              - The AML opcode
164  *
165  * RETURN:      A pointer to the name of the opcode (ASCII String)
166  *              Note: Never returns NULL.
167  *
168  * DESCRIPTION: Translate an opcode into a human-readable string
169  *
170  *****************************************************************************/

172 char *
173 AcpiPsGetOpcodeName (
174     UINT16                  Opcode)
175 {
176 #if defined(ACPI_DISASSEMBLER) || defined (ACPI_DEBUG_OUTPUT)

178     const ACPI_OPCODE_INFO  *Op;


181     Op = AcpiPsGetOpcodeInfo (Opcode);

183     /* Always guaranteed to return a valid pointer */

185     return (Op->Name);

187 #else
188     return ("OpcodeName unavailable");

190 #endif
191 }
```



```
194 /******************************************************************************
195  *
196  * FUNCTION:    AcpiPsGetArgumentCount
197  *
198  * PARAMETERS:  OpType              - Type associated with the AML opcode
199  *
200  * RETURN:      Argument count
201  *
202  * DESCRIPTION: Obtain the number of expected arguments for an AML opcode
203  *
204  *****************************************************************************/

206 UINT8
207 AcpiPsGetArgumentCount (
208     UINT32                  OpType)
209 {

211     if (OpType <= AML_TYPE_EXEC_6A_0T_1R)
212     {
213         return (AcpiGbl_ArgumentCount[OpType]);
214     }

216     return (0);
217 }


220 /*
221  * This table is directly indexed by the opcodes It returns
222  * an index into the opcode table (AcpiGbl_AmlOpInfo)
223  */
224 const UINT8 AcpiGbl_ShortOpIndex[256] =
225 {
226 /*              0     1     2     3     4     5     6     7  */
227 /*              8     9     A     B     C     D     E     F  */
228 /* 0x00 */    0x00, 0x01, _UNK, _UNK, _UNK, _UNK, 0x02, _UNK,
229 /* 0x08 */    0x03, _UNK, 0x04, 0x05, 0x06, 0x07, 0x6E, _UNK,
230 /* 0x10 */    0x08, 0x09, 0x0a, 0x6F, 0x0b, _UNK, _UNK, _UNK,
231 /* 0x18 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
232 /* 0x20 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
233 /* 0x28 */    _UNK, _UNK, _UNK, _UNK, _UNK, 0x63, _PFX, _PFX,
234 /* 0x30 */    0x67, 0x66, 0x68, 0x65, 0x69, 0x64, 0x6A, 0x7D,
235 /* 0x38 */    0x7F, 0x80, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
236 /* 0x40 */    _UNK, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
237 /* 0x48 */    _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
238 /* 0x50 */    _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
239 /* 0x58 */    _ASC, _ASC, _ASC, _UNK, _PFX, _UNK, _PFX, _ASC,
240 /* 0x60 */    0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13,
241 /* 0x68 */    0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, _UNK,
242 /* 0x70 */    0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x20, 0x21, 0x22,
243 /* 0x78 */    0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2a,
244 /* 0x80 */    0x2b, 0x2c, 0x2d, 0x2e, 0x70, 0x71, 0x2f, 0x30,
245 /* 0x88 */    0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x72,
246 /* 0x90 */    0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x73, 0x74,
247 /* 0x98 */    0x75, 0x76, _UNK, _UNK, 0x77, 0x78, 0x79, 0x7A,
248 /* 0xA0 */    0x3e, 0x3f, 0x40, 0x41, 0x42, 0x43, 0x60, 0x61,
249 /* 0xA8 */    0x62, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
250 /* 0xB0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
251 /* 0xB8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
252 /* 0xC0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
253 /* 0xC8 */    _UNK, _UNK, _UNK, _UNK, 0x44, _UNK, _UNK, _UNK,
254 /* 0xD0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
255 /* 0xD8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
256 /* 0xE0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
257 /* 0xE8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
258 /* 0xF0 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
```

```
/* 0xF8 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x45,
};

/*
 * This table is indexed by the second opcode of the extended opcode
 * pair. It returns an index into the opcode table (AcpiGbl_AmlOpInfo)
 */
const UINT8 AcpiGbl_LongOpIndex[NUM_EXTENDED_OPCODE] =
{
/*              0     1     2     3     4     5     6     7  */
/*              8     9     A     B     C     D     E     F  */
/* 0x00 */    _UNK, 0x46, 0x47, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x08 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x10 */    _UNK, _UNK, 0x48, 0x49, _UNK, _UNK, _UNK, _UNK,
/* 0x18 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x7B,
/* 0x20 */    0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51,
/* 0x28 */    0x52, 0x53, 0x54, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x30 */    0x55, 0x56, 0x57, 0x7e, _UNK, _UNK, _UNK, _UNK,
/* 0x38 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x40 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x48 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x50 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x58 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x60 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x68 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x70 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x78 */    _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
/* 0x80 */    0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
/* 0x88 */    0x7C,
};
```

****
21770 Thu Dec 26 13:49:25 2013
new/usr/src/common/acpica/components/parser/psparse.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: psparse - Parser top level AML parse routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


/*
 * Parse the AML and build an operation tree as most interpreters,
 * like Perl, do. Parsing is done by hand rather than with a YACC
 * generated parser to tightly constrain stack and dynamic memory
 * usage. At the same time, parsing is kept flexible and the code
 * fairly compact by parsing based on a list of AML opcode
 * templates in AmlOpInfo[]
 */

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acdispat.h"
#include "amlcode.h"
#include "acinterp.h"
```



```
#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psparse")


/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetOpcodeSize
 *
 * PARAMETERS:  Opcode          - An AML opcode
 *
 * RETURN:      Size of the opcode, in bytes (1 or 2)
 *
 * DESCRIPTION: Get the size of the current opcode.
 *
 ******************************************************************************/

UINT32
AcpiPsGetOpcodeSize (
    UINT32                  Opcode)
{

    /* Extended (2-byte) opcode if > 255 */

    if (Opcode > 0x00FF)
    {
        return (2);
    }

    /* Otherwise, just a single byte opcode */

    return (1);
}
```

unchanged_portion_omitted_

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiPsCompleteThisOp
 *
 * PARAMETERS:  WalkState       - Current State
 *              Op              - Op to complete
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Perform any cleanup at the completion of an Op.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiPsCompleteThisOp (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Prev;
    ACPI_PARSE_OBJECT       *Next;
    const ACPI_OPCODE_INFO  *ParentInfo;
    ACPI_PARSE_OBJECT       *ReplacementOp = NULL;
    ACPI_STATUS             Status = AE_OK;


    ACPI_FUNCTION_TRACE_PTR (PsCompleteThisOp, Op);


    /* Check for null Op, can happen if AML code is corrupt */
```

```
160     if (!Op)
161     {
162         return_ACPI_STATUS (AE_OK);  /* OK for now */
163     }

165     /* Delete this op and the subtree below it if asked to */

167     if (((WalkState->ParseFlags & ACPI_PARSE_TREE_MASK) != ACPI_PARSE_DELETE_TRE
168          (WalkState->OpInfo->Class == AML_CLASS_ARGUMENT))
169     {
170         return_ACPI_STATUS (AE_OK);
171     }

173     /* Make sure that we only delete this subtree */

175     if (Op->Common.Parent)
176     {
177         Prev = Op->Common.Parent->Common.Value.Arg;
178         if (!Prev)
179         {
180             /* Nothing more to do */

182             goto Cleanup;
183         }

185         /*
186          * Check if we need to replace the operator and its subtree
187          * with a return value op (placeholder op)
188          */
189         ParentInfo = AcpiPsGetOpcodeInfo (Op->Common.Parent->Common.AmlOpcode);

191         switch (ParentInfo->Class)
192         {
193         case AML_CLASS_CONTROL:

195             break;

197         case AML_CLASS_CREATE:

198             /*
199              * These opcodes contain TermArg operands. The current
200              * op must be replaced by a placeholder return op
201              */
202             ReplacementOp = AcpiPsAllocOp (AML_INT_RETURN_VALUE_OP);
203             if (!ReplacementOp)
204             {
205                 Status = AE_NO_MEMORY;
206             }
207             break;

209         case AML_CLASS_NAMED_OBJECT:

210             /*
211              * These opcodes contain TermArg operands. The current
212              * op must be replaced by a placeholder return op
213              */
214             if ((Op->Common.Parent->Common.AmlOpcode == AML_REGION_OP)       ||
215                 (Op->Common.Parent->Common.AmlOpcode == AML_DATA_REGION_OP)  ||
216                 (Op->Common.Parent->Common.AmlOpcode == AML_BUFFER_OP)       ||
217                 (Op->Common.Parent->Common.AmlOpcode == AML_PACKAGE_OP)      ||
218                 (Op->Common.Parent->Common.AmlOpcode == AML_BANK_FIELD_OP)   ||
219                 (Op->Common.Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
220             {
221                 ReplacementOp = AcpiPsAllocOp (AML_INT_RETURN_VALUE_OP);
222                 if (!ReplacementOp)
223                 {
```



```
224                     Status = AE_NO_MEMORY;
225                 }
226             }
227             else if ((Op->Common.Parent->Common.AmlOpcode == AML_NAME_OP) &&
228                      (WalkState->PassNumber <= ACPI_IMODE_LOAD_PASS2))
229             {
230                 if ((Op->Common.AmlOpcode == AML_BUFFER_OP) ||
231                     (Op->Common.AmlOpcode == AML_PACKAGE_OP) ||
232                     (Op->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
233                 {
234                     ReplacementOp = AcpiPsAllocOp (Op->Common.AmlOpcode);
235                     if (!ReplacementOp)
236                     {
237                         Status = AE_NO_MEMORY;
238                     }
239                     else
240                     {
241                         ReplacementOp->Named.Data = Op->Named.Data;
242                         ReplacementOp->Named.Length = Op->Named.Length;
243                     }
244                 }
245             }
246             break;

248         default:

250             ReplacementOp = AcpiPsAllocOp (AML_INT_RETURN_VALUE_OP);
251             if (!ReplacementOp)
252             {
253                 Status = AE_NO_MEMORY;
254             }
255         }

257         /* We must unlink this op from the parent tree */

259         if (Prev == Op)
260         {
261             /* This op is the first in the list */

263             if (ReplacementOp)
264             {
265                 ReplacementOp->Common.Parent        = Op->Common.Parent;
266                 ReplacementOp->Common.Value.Arg     = NULL;
267                 ReplacementOp->Common.Node          = Op->Common.Node;
268                 Op->Common.Parent->Common.Value.Arg = ReplacementOp;
269                 ReplacementOp->Common.Next          = Op->Common.Next;
270             }
271             else
272             {
273                 Op->Common.Parent->Common.Value.Arg = Op->Common.Next;
274             }
275         }

277         /* Search the parent list */

279         else while (Prev)
280         {
281             /* Traverse all siblings in the parent's argument list */

283             Next = Prev->Common.Next;
284             if (Next == Op)
285             {
286                 if (ReplacementOp)
287                 {
288                     ReplacementOp->Common.Parent    = Op->Common.Parent;
289                     ReplacementOp->Common.Value.Arg = NULL;
```

```
290                     ReplacementOp->Common.Node      = Op->Common.Node;
291                     Prev->Common.Next               = ReplacementOp;
292                     ReplacementOp->Common.Next      = Op->Common.Next;
293                     Next = NULL;
294                 }
295                 else
296                 {
297                     Prev->Common.Next = Op->Common.Next;
298                     Next = NULL;
299                 }
300             }
301             Prev = Next;
302         }
303     }


306 Cleanup:

308     /* Now we can actually delete the subtree rooted at Op */

310     AcpiPsDeleteParseTree (Op);
311     return_ACPI_STATUS (Status);
312 }


315 /*******************************************************************************
316  *
317  * FUNCTION:    AcpiPsNextParseState
318  *
319  * PARAMETERS:  WalkState           - Current state
320  *              Op                  - Current parse op
321  *              CallbackStatus      - Status from previous operation
322  *
323  * RETURN:      Status
324  *
325  * DESCRIPTION: Update the parser state based upon the return exception from
326  *              the parser callback.
327  *
328  ******************************************************************************/

330 ACPI_STATUS
331 AcpiPsNextParseState (
332     ACPI_WALK_STATE         *WalkState,
333     ACPI_PARSE_OBJECT       *Op,
334     ACPI_STATUS             CallbackStatus)
335 {
336     ACPI_PARSE_STATE        *ParserState = &WalkState->ParserState;
337     ACPI_STATUS             Status = AE_CTRL_PENDING;


340     ACPI_FUNCTION_TRACE_PTR (PsNextParseState, Op);


343     switch (CallbackStatus)
344     {
345     case AE_CTRL_TERMINATE:
346         /*
347          * A control method was terminated via a RETURN statement.
348          * The walk of this method is complete.
349          */
350         ParserState->Aml = ParserState->AmlEnd;
351         Status = AE_CTRL_TERMINATE;
352         break;


354     case AE_CTRL_BREAK:
```



```
356         ParserState->Aml = WalkState->AmlLastWhile;
357         WalkState->ControlState->Common.Value = FALSE;
358         Status = AE_CTRL_BREAK;
359         break;


361     case AE_CTRL_CONTINUE:

363         ParserState->Aml = WalkState->AmlLastWhile;
364         Status = AE_CTRL_CONTINUE;
365         break;


367     case AE_CTRL_PENDING:

369         ParserState->Aml = WalkState->AmlLastWhile;
370         break;

372 #if 0
373     case AE_CTRL_SKIP:

375         ParserState->Aml = ParserState->Scope->ParseScope.PkgEnd;
376         Status = AE_OK;
377         break;
378 #endif

380     case AE_CTRL_TRUE:
381         /*
382          * Predicate of an IF was true, and we are at the matching ELSE.
383          * Just close out this package
384          */
385         ParserState->Aml = AcpiPsGetNextPackageEnd (ParserState);
386         Status = AE_CTRL_PENDING;
387         break;


389     case AE_CTRL_FALSE:
390         /*
391          * Either an IF/WHILE Predicate was false or we encountered a BREAK
392          * opcode. In both cases, we do not execute the rest of the
393          * package;  We simply close out the parent (finishing the walk of
394          * this branch of the tree) and continue execution at the parent
395          * level.
396          */
397         ParserState->Aml = ParserState->Scope->ParseScope.PkgEnd;

399         /* In the case of a BREAK, just force a predicate (if any) to FALSE */

401         WalkState->ControlState->Common.Value = FALSE;
402         Status = AE_CTRL_END;
403         break;


405     case AE_CTRL_TRANSFER:

407         /* A method call (invocation) -- transfer control */

409         Status = AE_CTRL_TRANSFER;
410         WalkState->PrevOp = Op;
411         WalkState->MethodCallOp = Op;
412         WalkState->MethodCallNode = (Op->Common.Value.Arg)->Common.Node;

414         /* Will return value (if any) be used by the caller? */

416         WalkState->ReturnUsed = AcpiDsIsResultUsed (Op, WalkState);
```

```
417         break;

419     default:

421         Status = CallbackStatus;
422         if ((CallbackStatus & AE_CODE_MASK) == AE_CODE_CONTROL)
423         {
424             Status = AE_OK;
425         }
426         break;
427     }

429     return_ACPI_STATUS (Status);
430 }
```

unchanged_portion_omitted_

****
8723 Thu Dec 26 13:49:26 2013
new/usr/src/common/acpica/components/parser/psscope.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: psscope - Parser scope stack management routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psscope")


/******************************************************************************
 *
 * FUNCTION:    AcpiPsGetParentScope
 *
 * PARAMETERS:  ParserState         - Current parser state object
 *
 * RETURN:      Pointer to an Op object
```



```
 *
 * DESCRIPTION: Get parent of current op being parsed
 *
 *****************************************************************************/

ACPI_PARSE_OBJECT *
AcpiPsGetParentScope (
    ACPI_PARSE_STATE        *ParserState)
{

    return (ParserState->Scope->ParseScope.Op);
}
```

unchanged_portion_omitted_

****
8614 Thu Dec 26 13:49:26 2013
new/usr/src/common/acpica/components/parser/pstree.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: pstree - Parser op tree manipulation/traversal/search
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __PSTREE_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("pstree")

/* Local prototypes */

#ifdef ACPI_OBSOLETE_FUNCTIONS
ACPI_PARSE_OBJECT *
AcpiPsGetChild (
```



```
    ACPI_PARSE_OBJECT       *op);
#endif


/******************************************************************************
 *
 * FUNCTION:    AcpiPsGetArg
 *
 * PARAMETERS:  Op              - Get an argument for this op
 *              Argn            - Nth argument to get
 *
 * RETURN:      The argument (as an Op object). NULL if argument does not exist
 *
 * DESCRIPTION: Get the specified op's argument.
 *
 *****************************************************************************/

ACPI_PARSE_OBJECT *
AcpiPsGetArg (
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  Argn)
{
    ACPI_PARSE_OBJECT       *Arg = NULL;
    const ACPI_OPCODE_INFO  *OpInfo;


    ACPI_FUNCTION_ENTRY ();

/*
    if (Op->Common.AmlOpcode == AML_INT_CONNECTION_OP)
    {
        return (Op->Common.Value.Arg);
    }
*/

    /* Get the info structure for this opcode */

    OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
    if (OpInfo->Class == AML_CLASS_UNKNOWN)
    {
        /* Invalid opcode or ASCII character */

        return (NULL);
    }

    /* Check if this opcode requires argument sub-objects */

    if (!(OpInfo->Flags & AML_HAS_ARGS))
    {
        /* Has no linked argument objects */

        return (NULL);
    }

    /* Get the requested argument object */

    Arg = Op->Common.Value.Arg;
    while (Arg && Argn)
    {
        Argn--;
        Arg = Arg->Common.Next;
    }

    return (Arg);
}
```

_____unchanged_portion_omitted_

```
#ifdef ACPI_OBSOLETE_FUNCTIONS
/*******************************************************************************
 *
 * FUNCTION:    AcpiPsGetChild
 *
 * PARAMETERS:  Op              - Get the child of this Op
 *
 * RETURN:      Child Op, Null if none is found.
 *
 * DESCRIPTION: Get op's children or NULL if none
 *
 ******************************************************************************/

ACPI_PARSE_OBJECT *
AcpiPsGetChild (
    ACPI_PARSE_OBJECT       *Op)
{
    ACPI_PARSE_OBJECT       *Child = NULL;


    ACPI_FUNCTION_ENTRY ();


    switch (Op->Common.AmlOpcode)
    {
    case AML_SCOPE_OP:
    case AML_ELSE_OP:
    case AML_DEVICE_OP:
    case AML_THERMAL_ZONE_OP:
    case AML_INT_METHODCALL_OP:

        Child = AcpiPsGetArg (Op, 0);
        break;


    case AML_BUFFER_OP:
    case AML_PACKAGE_OP:
    case AML_METHOD_OP:
    case AML_IF_OP:
    case AML_WHILE_OP:
    case AML_FIELD_OP:

        Child = AcpiPsGetArg (Op, 1);
        break;


    case AML_POWER_RES_OP:
    case AML_INDEX_FIELD_OP:

        Child = AcpiPsGetArg (Op, 2);
        break;


    case AML_PROCESSOR_OP:
    case AML_BANK_FIELD_OP:

        Child = AcpiPsGetArg (Op, 3);
        break;

    default:

    default:
        /* All others have no children */
```



```
        break;
    }

    return (Child);
}
#endif
```

```
**********************************************************
    7255 Thu Dec 26 13:49:26 2013
new/usr/src/common/acpica/components/parser/psutils.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: psutils - Parser miscellaneous utilities (Parser only)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psutils")


/******************************************************************************
 *
 * FUNCTION:    AcpiPsCreateScopeOp
 *
 * PARAMETERS:  None
 *
```



```
 * RETURN:      A new Scope object, null on failure
 *
 * DESCRIPTION: Create a Scope and associated namepath op with the root name
 *
 *****************************************************************************/

ACPI_PARSE_OBJECT *
AcpiPsCreateScopeOp (
    void)
{
    ACPI_PARSE_OBJECT       *ScopeOp;


    ScopeOp = AcpiPsAllocOp (AML_SCOPE_OP);
    if (!ScopeOp)
    {
        return (NULL);
    }

    ScopeOp->Named.Name = ACPI_ROOT_NAME;
    return (ScopeOp);
}
______unchanged_portion_omitted_


/*
 * Is "c" a namestring prefix character?
 */
BOOLEAN
AcpiPsIsPrefixChar (
    UINT32                  c)
{
    return ((BOOLEAN) (c == '\\' || c == '^'));
}


/*
 * Get op's name (4-byte name segment) or 0 if unnamed
 */
UINT32
AcpiPsGetName (
    ACPI_PARSE_OBJECT       *Op)
{

    /* The "generic" object has no name associated with it */

    if (Op->Common.Flags & ACPI_PARSEOP_GENERIC)
    {
        return (0);
    }

    /* Only the "Extended" parse objects have a name */

    return (Op->Named.Name);
}
______unchanged_portion_omitted_
```

****
3794 Thu Dec 26 13:49:27 2013
new/usr/src/common/acpica/components/parser/pswalk.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: pswalk - Parser routines to walk parsed op tree(s)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpi.h"
#include "accommon.h"
#include "acparser.h"

#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("pswalk")


/******************************************************************************
 *
 * FUNCTION:    AcpiPsDeleteParseTree
 *
 * PARAMETERS:  SubtreeRoot         - Root of tree (or subtree) to delete
 *
 * RETURN:      None
```



```
 *
 * DESCRIPTION: Delete a portion of or an entire parse tree.
 *
 *****************************************************************************/

void
AcpiPsDeleteParseTree (
    ACPI_PARSE_OBJECT       *SubtreeRoot)
{
    ACPI_PARSE_OBJECT       *Op = SubtreeRoot;
    ACPI_PARSE_OBJECT       *Next = NULL;
    ACPI_PARSE_OBJECT       *Parent = NULL;


    ACPI_FUNCTION_TRACE_PTR (PsDeleteParseTree, SubtreeRoot);


    /* Visit all nodes in the subtree */

    while (Op)
    {
        /* Check if we are not ascending */

        if (Op != Parent)
        {
            /* Look for an argument or child of the current op */

            Next = AcpiPsGetArg (Op, 0);
            if (Next)
            {
                /* Still going downward in tree (Op is not completed yet) */

                Op = Next;
                continue;
            }
        }

        /* No more children, this Op is complete. */

        Next = Op->Common.Next;
        Parent = Op->Common.Parent;

        AcpiPsFreeOp (Op);

        /* If we are back to the starting point, the walk is complete. */

        if (Op == SubtreeRoot)
        {
            return_VOID;
        }
        if (Next)
        {
            Op = Next;
        }
        else
        {
            Op = Parent;
        }
    }

    return_VOID;
}
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   11999 Thu Dec 26 13:49:27 2013
new/usr/src/common/acpica/components/parser/psxface.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: psxface - Parser external interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __PSXFACE_C__

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "acdispat.h"
#include "acinterp.h"
#include "actables.h"


#define _COMPONENT          ACPI_PARSER
        ACPI_MODULE_NAME    ("psxface")

/* Local Prototypes */

static void
AcpiPsStartTrace (
    ACPI_EVALUATE_INFO      *Info);

static void
AcpiPsStopTrace (
    ACPI_EVALUATE_INFO      *Info);

static void
AcpiPsUpdateParameterList (
    ACPI_EVALUATE_INFO      *Info,
    UINT16                  Action);


/******************************************************************************
 *
 * FUNCTION:    AcpiDebugTrace
 *
 * PARAMETERS:  MethodName      - Valid ACPI name string
 *              DebugLevel      - Optional level mask. 0 to use default
 *              DebugLayer      - Optional layer mask. 0 to use default
 *              Flags           - bit 1: one shot(1) or persistent(0)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: External interface to enable debug tracing during control
 *              method execution
 *
 *****************************************************************************/

ACPI_STATUS
AcpiDebugTrace (
    char                    *Name,
    UINT32                  DebugLevel,
    UINT32                  DebugLayer,
    UINT32                  Flags)
{
    ACPI_STATUS             Status;


    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* TBDs: Validate name, allow full path or just nameseg */

    AcpiGbl_TraceMethodName = *ACPI_CAST_PTR (UINT32, Name);
    AcpiGbl_TraceFlags = Flags;

    if (DebugLevel)
    {
        AcpiGbl_TraceDbgLevel = DebugLevel;
    }
    if (DebugLayer)
    {
        AcpiGbl_TraceDbgLayer = DebugLayer;
    }

    (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    return (AE_OK);
}


/******************************************************************************
 *
```

```
 126  * FUNCTION:    AcpiPsStartTrace
 127  *
 128  * PARAMETERS:  Info        - Method info struct
 129  *
 130  * RETURN:      None
 131  *
 132  * DESCRIPTION: Start control method execution trace
 133  *
 134  ******************************************************************************/

 136 static void
 137 AcpiPsStartTrace (
 138     ACPI_EVALUATE_INFO      *Info)
 139 {
 140     ACPI_STATUS             Status;


 143     ACPI_FUNCTION_ENTRY ();


 146     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
 147     if (ACPI_FAILURE (Status))
 148     {
 149         return;
 150     }

 152     if ((!AcpiGbl_TraceMethodName) ||
 153         (AcpiGbl_TraceMethodName != Info->Node->Name.Integer))
 153         (AcpiGbl_TraceMethodName != Info->ResolvedNode->Name.Integer))
 154     {
 155         goto Exit;
 156     }

 158     AcpiGbl_OriginalDbgLevel = AcpiDbgLevel;
 159     AcpiGbl_OriginalDbgLayer = AcpiDbgLayer;

 161     AcpiDbgLevel = 0x00FFFFFF;
 162     AcpiDbgLayer = ACPI_UINT32_MAX;

 164     if (AcpiGbl_TraceDbgLevel)
 165     {
 166         AcpiDbgLevel = AcpiGbl_TraceDbgLevel;
 167     }
 168     if (AcpiGbl_TraceDbgLayer)
 169     {
 170         AcpiDbgLayer = AcpiGbl_TraceDbgLayer;
 171     }


 174 Exit:
 175     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
 176 }


 179 /******************************************************************************
 180  *
 181  * FUNCTION:    AcpiPsStopTrace
 182  *
 183  * PARAMETERS:  Info        - Method info struct
 184  *
 185  * RETURN:      None
 186  *
 187  * DESCRIPTION: Stop control method execution trace
 188  *
 189  ******************************************************************************/
```



```
 191 static void
 192 AcpiPsStopTrace (
 193     ACPI_EVALUATE_INFO      *Info)
 194 {
 195     ACPI_STATUS             Status;


 198     ACPI_FUNCTION_ENTRY ();


 201     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
 202     if (ACPI_FAILURE (Status))
 203     {
 204         return;
 205     }

 207     if ((!AcpiGbl_TraceMethodName) ||
 208         (AcpiGbl_TraceMethodName != Info->Node->Name.Integer))
 208         (AcpiGbl_TraceMethodName != Info->ResolvedNode->Name.Integer))
 209     {
 210         goto Exit;
 211     }

 213     /* Disable further tracing if type is one-shot */

 215     if (AcpiGbl_TraceFlags & 1)
 216     {
 217         AcpiGbl_TraceMethodName = 0;
 218         AcpiGbl_TraceDbgLevel = 0;
 219         AcpiGbl_TraceDbgLayer = 0;
 220     }

 222     AcpiDbgLevel = AcpiGbl_OriginalDbgLevel;
 223     AcpiDbgLayer = AcpiGbl_OriginalDbgLayer;

 225 Exit:
 226     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
 227 }


 230 /******************************************************************************
 231  *
 232  * FUNCTION:    AcpiPsExecuteMethod
 233  *
 234  * PARAMETERS:  Info            - Method info block, contains:
 235  *                  Node            - Method Node to execute
 236  *                  ObjDesc         - Method object
 237  *                  Parameters      - List of parameters to pass to the method,
 238  *                                    terminated by NULL. Params itself may be
 239  *                                    NULL if no parameters are being passed.
 240  *                  ReturnObject    - Where to put method's return value (if
 241  *                                    any). If NULL, no value is returned.
 242  *                  ParameterType   - Type of Parameter list
 243  *                  ReturnObject    - Where to put method's return value (if
 244  *                                    any). If NULL, no value is returned.
 245  *                  PassNumber      - Parse or execute pass
 246  *
 247  * RETURN:      Status
 248  *
 249  * DESCRIPTION: Execute a control method
 250  *
 251  ******************************************************************************/

 253 ACPI_STATUS
 254 AcpiPsExecuteMethod (
 255     ACPI_EVALUATE_INFO      *Info)
```

```
256 {
257     ACPI_STATUS             Status;
258     ACPI_PARSE_OBJECT       *Op;
259     ACPI_WALK_STATE         *WalkState;


262     ACPI_FUNCTION_TRACE (PsExecuteMethod);


265     /* Quick validation of DSDT header */

267     AcpiTbCheckDsdtHeader ();

269     /* Validate the Info and method Node */

271     if (!Info || !Info->Node)
271     if (!Info || !Info->ResolvedNode)
272     {
273         return_ACPI_STATUS (AE_NULL_ENTRY);
274     }

276     /* Init for new method, wait on concurrency semaphore */

278     Status = AcpiDsBeginMethodExecution (Info->Node, Info->ObjDesc, NULL);
278     Status = AcpiDsBeginMethodExecution (Info->ResolvedNode, Info->ObjDesc, NULL
279     if (ACPI_FAILURE (Status))
280     {
281         return_ACPI_STATUS (Status);
282     }

284     /*
285      * The caller "owns" the parameters, so give each one an extra reference
286      */
287     AcpiPsUpdateParameterList (Info, REF_INCREMENT);

289     /* Begin tracing if requested */

291     AcpiPsStartTrace (Info);

293     /*
294      * Execute the method. Performs parse simultaneously
295      */
296     ACPI_DEBUG_PRINT ((ACPI_DB_PARSE,
297         "**** Begin Method Parse/Execute [%4.4s] **** Node=%p Obj=%p\n",
298         Info->Node->Name.Ascii, Info->Node, Info->ObjDesc));
298         Info->ResolvedNode->Name.Ascii, Info->ResolvedNode, Info->ObjDesc));

300     /* Create and init a Root Node */

302     Op = AcpiPsCreateScopeOp ();
303     if (!Op)
304     {
305         Status = AE_NO_MEMORY;
306         goto Cleanup;
307     }

309     /* Create and initialize a new walk state */

311     Info->PassNumber = ACPI_IMODE_EXECUTE;
312     WalkState = AcpiDsCreateWalkState (
313                     Info->ObjDesc->Method.OwnerId, NULL, NULL, NULL);
314     if (!WalkState)
315     {
316         Status = AE_NO_MEMORY;
317         goto Cleanup;
318     }
```



```
320     Status = AcpiDsInitAmlWalk (WalkState, Op, Info->Node,
320     Status = AcpiDsInitAmlWalk (WalkState, Op, Info->ResolvedNode,
321                 Info->ObjDesc->Method.AmlStart,
322                 Info->ObjDesc->Method.AmlLength, Info, Info->PassNumber);
323     if (ACPI_FAILURE (Status))
324     {
325         AcpiDsDeleteWalkState (WalkState);
326         goto Cleanup;
327     }

329     if (Info->ObjDesc->Method.InfoFlags & ACPI_METHOD_MODULE_LEVEL)
330     {
331         WalkState->ParseFlags |= ACPI_PARSE_MODULE_LEVEL;
332     }

334     /* Invoke an internal method if necessary */

336     if (Info->ObjDesc->Method.InfoFlags & ACPI_METHOD_INTERNAL_ONLY)
337     {
338         Status = Info->ObjDesc->Method.Dispatch.Implementation (WalkState);
339         Info->ReturnObject = WalkState->ReturnDesc;

341         /* Cleanup states */

343         AcpiDsScopeStackClear (WalkState);
344         AcpiPsCleanupScope (&WalkState->ParserState);
345         AcpiDsTerminateControlMethod (WalkState->MethodDesc, WalkState);
346         AcpiDsDeleteWalkState (WalkState);
347         goto Cleanup;
348     }

350     /*
351      * Start method evaluation with an implicit return of zero.
352      * This is done for Windows compatibility.
351      * Start method evaluation with an implicit return of zero. This is done
352      * for Windows compatibility.
353      */
354     if (AcpiGbl_EnableInterpreterSlack)
355     {
356         WalkState->ImplicitReturnObj =
357             AcpiUtCreateIntegerObject ((UINT64) 0);
358         if (!WalkState->ImplicitReturnObj)
359         {
360             Status = AE_NO_MEMORY;
361             AcpiDsDeleteWalkState (WalkState);
362             goto Cleanup;
363         }
364     }

366     /* Parse the AML */

368     Status = AcpiPsParseAml (WalkState);

370     /* WalkState was deleted by ParseAml */

372 Cleanup:
373     AcpiPsDeleteParseTree (Op);

375     /* End optional tracing */

377     AcpiPsStopTrace (Info);

379     /* Take away the extra reference that we gave the parameters above */

381     AcpiPsUpdateParameterList (Info, REF_DECREMENT);
```

```
383     /* Exit now if error above */

385     if (ACPI_FAILURE (Status))
386     {
387         return_ACPI_STATUS (Status);
388     }

390     /*
391      * If the method has returned an object, signal this to the caller with
392      * a control exception code
393      */
394     if (Info->ReturnObject)
395     {
396         ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "Method returned ObjDesc=%p\n",
397             Info->ReturnObject));
398         ACPI_DUMP_STACK_ENTRY (Info->ReturnObject);

400         Status = AE_CTRL_RETURN_VALUE;
401     }

403     return_ACPI_STATUS (Status);
404 }
```

unchanged_portion_omitted_

```
**********************************************************
   13908 Thu Dec 26 13:49:28 2013
new/usr/src/common/acpica/components/resources/rsaddr.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsaddr - Address resource descriptors (16/32/64)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSADDR_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsaddr")


/******************************************************************************
 *
 * AcpiRsConvertAddress16 - All WORD (16-bit) address resources
 *
 *****************************************************************************/
```



```
ACPI_RSCONVERT_INFO     AcpiRsConvertAddress16[5] =
{
    {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_ADDRESS16,
                        ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS16),
                        ACPI_RSC_TABLE_SIZE (AcpiRsConvertAddress16)},

    {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_ADDRESS16,
                        sizeof (AML_RESOURCE_ADDRESS16),
                        0},

    /* Resource Type, General Flags, and Type-Specific Flags */

    {ACPI_RSC_ADDRESS,  0, 0, 0},

    /*
     * These fields are contiguous in both the source and destination:
     * Address Granularity
     * Address Range Minimum
     * Address Range Maximum
     * Address Translation Offset
     * Address Length
     */
    {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.Address16.Granularity),
                        AML_OFFSET (Address16.Granularity),
                        5},

    /* Optional ResourceSource (Index and String) */

    {ACPI_RSC_SOURCE,   ACPI_RS_OFFSET (Data.Address16.ResourceSource),
                        0,
                        sizeof (AML_RESOURCE_ADDRESS16)}
};
```

unchanged_portion_omitted_

```
**********************************************************
   23607 Thu Dec 26 13:49:28 2013
new/usr/src/common/acpica/components/resources/rscalc.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: rscalc - Calculate stream and list lengths
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __RSCALC_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acresrc.h"
  49 #include "acnamesp.h"


  52 #define _COMPONENT          ACPI_RESOURCES
  53         ACPI_MODULE_NAME    ("rscalc")


  56 /* Local prototypes */

  58 static UINT8
```



```
  59 AcpiRsCountSetBits (
  60     UINT16                  BitField);

  62 static ACPI_RS_LENGTH
  63 AcpiRsStructOptionLength (
  64     ACPI_RESOURCE_SOURCE    *ResourceSource);

  66 static UINT32
  67 AcpiRsStreamOptionLength (
  68     UINT32                  ResourceLength,
  69     UINT32                  MinimumTotalLength);


  72 /******************************************************************************
  73  *
  74  * FUNCTION:    AcpiRsCountSetBits
  75  *
  76  * PARAMETERS:  BitField        - Field in which to count bits
  77  *
  78  * RETURN:      Number of bits set within the field
  79  *
  80  * DESCRIPTION: Count the number of bits set in a resource field. Used for
  81  *              (Short descriptor) interrupt and DMA lists.
  82  *
  83  *****************************************************************************/

  85 static UINT8
  86 AcpiRsCountSetBits (
  87     UINT16                  BitField)
  88 {
  89     UINT8                   BitsSet;


  92     ACPI_FUNCTION_ENTRY ();


  95     for (BitsSet = 0; BitField; BitsSet++)
  96     {
  97         /* Zero the least significant bit that is set */

  99         BitField &= (UINT16) (BitField - 1);
 100     }

 102     return (BitsSet);
 103 }
______unchanged_portion_omitted_


 195 /******************************************************************************
 196  *
 197  * FUNCTION:    AcpiRsGetAmlLength
 198  *
 199  * PARAMETERS:  Resource            - Pointer to the resource linked list
 200  *              ResourceListSize    - Size of the resource linked list
 201  *              SizeNeeded          - Where the required size is returned
 202  *
 203  * RETURN:      Status
 204  *
 205  * DESCRIPTION: Takes a linked list of internal resource descriptors and
 206  *              calculates the size buffer needed to hold the corresponding
 207  *              external resource byte stream.
 208  *
 209  *****************************************************************************/

 211 ACPI_STATUS
 212 AcpiRsGetAmlLength (
```

```
213     ACPI_RESOURCE           *Resource,
214     ACPI_SIZE               ResourceListSize,
215     ACPI_SIZE               *SizeNeeded)
216 {
217     ACPI_SIZE               AmlSizeNeeded = 0;
218     ACPI_RESOURCE           *ResourceEnd;
219     ACPI_RS_LENGTH          TotalSize;


222     ACPI_FUNCTION_TRACE (RsGetAmlLength);


225     /* Traverse entire list of internal resource descriptors */

227     ResourceEnd = ACPI_ADD_PTR (ACPI_RESOURCE, Resource, ResourceListSize);
228     while (Resource < ResourceEnd)
224     while (Resource)
229     {
230         /* Validate the descriptor type */

232         if (Resource->Type > ACPI_RESOURCE_TYPE_MAX)
233         {
234             return_ACPI_STATUS (AE_AML_INVALID_RESOURCE_TYPE);
235         }

237         /* Sanity check the length. It must not be zero, or we loop forever */

239         if (!Resource->Length)
240         {
241             return_ACPI_STATUS (AE_AML_BAD_RESOURCE_LENGTH);
242         }

244         /* Get the base size of the (external stream) resource descriptor */

246         TotalSize = AcpiGbl_AmlResourceSizes [Resource->Type];

248         /*
249          * Augment the base size for descriptors with optional and/or
250          * variable-length fields
251          */
252         switch (Resource->Type)
253         {
254         case ACPI_RESOURCE_TYPE_IRQ:

256             /* Length can be 3 or 2 */

258             if (Resource->Data.Irq.DescriptorLength == 2)
259             {
260                 TotalSize--;
261             }
262             break;


265         case ACPI_RESOURCE_TYPE_START_DEPENDENT:

267             /* Length can be 1 or 0 */

269             if (Resource->Data.Irq.DescriptorLength == 0)
270             {
271                 TotalSize--;
272             }
273             break;


276         case ACPI_RESOURCE_TYPE_VENDOR:
277             /*
```



```
278              * Vendor Defined Resource:
279              * For a Vendor Specific resource, if the Length is between 1 and 7
280              * it will be created as a Small Resource data type, otherwise it
281              * is a Large Resource data type.
282              */
283             if (Resource->Data.Vendor.ByteLength > 7)
284             {
285                 /* Base size of a Large resource descriptor */

287                 TotalSize = sizeof (AML_RESOURCE_LARGE_HEADER);
288             }

290             /* Add the size of the vendor-specific data */

292             TotalSize = (ACPI_RS_LENGTH)
293                 (TotalSize + Resource->Data.Vendor.ByteLength);
294             break;


297         case ACPI_RESOURCE_TYPE_END_TAG:
298             /*
299              * End Tag:
300              * We are done -- return the accumulated total size.
301              */
302             *SizeNeeded = AmlSizeNeeded + TotalSize;

304             /* Normal exit */

306             return_ACPI_STATUS (AE_OK);


309         case ACPI_RESOURCE_TYPE_ADDRESS16:
310             /*
311              * 16-Bit Address Resource:
312              * Add the size of the optional ResourceSource info
313              */
314             TotalSize = (ACPI_RS_LENGTH)
315                 (TotalSize + AcpiRsStructOptionLength (
316                                 &Resource->Data.Address16.ResourceSource));
317             break;


320         case ACPI_RESOURCE_TYPE_ADDRESS32:
321             /*
322              * 32-Bit Address Resource:
323              * Add the size of the optional ResourceSource info
324              */
325             TotalSize = (ACPI_RS_LENGTH)
326                 (TotalSize + AcpiRsStructOptionLength (
327                                 &Resource->Data.Address32.ResourceSource));
328             break;


331         case ACPI_RESOURCE_TYPE_ADDRESS64:
332             /*
333              * 64-Bit Address Resource:
334              * Add the size of the optional ResourceSource info
335              */
336             TotalSize = (ACPI_RS_LENGTH)
337                 (TotalSize + AcpiRsStructOptionLength (
338                                 &Resource->Data.Address64.ResourceSource));
339             break;


342         case ACPI_RESOURCE_TYPE_EXTENDED_IRQ:
343             /*
```

```
344              * Extended IRQ Resource:
345              * Add the size of each additional optional interrupt beyond the
346              * required 1 (4 bytes for each UINT32 interrupt number)
347              */
348             TotalSize = (ACPI_RS_LENGTH)
349                 (TotalSize +
350                 ((Resource->Data.ExtendedIrq.InterruptCount - 1) * 4) +

352                 /* Add the size of the optional ResourceSource info */

354                 AcpiRsStructOptionLength (
355                     &Resource->Data.ExtendedIrq.ResourceSource));
356             break;


359         case ACPI_RESOURCE_TYPE_GPIO:

361             TotalSize = (ACPI_RS_LENGTH) (TotalSize + (Resource->Data.Gpio.PinTa
362                 Resource->Data.Gpio.ResourceSource.StringLength +
363                 Resource->Data.Gpio.VendorLength);

365             break;


368         case ACPI_RESOURCE_TYPE_SERIAL_BUS:

370             TotalSize = AcpiGbl_AmlResourceSerialBusSizes [Resource->Data.Common

372             TotalSize = (ACPI_RS_LENGTH) (TotalSize +
373                 Resource->Data.I2cSerialBus.ResourceSource.StringLength +
374                 Resource->Data.I2cSerialBus.VendorLength);

376             break;

378         default:

380             break;
381         }

383         /* Update the total */

385         AmlSizeNeeded += TotalSize;

387         /* Point to the next object */

389         Resource = ACPI_ADD_PTR (ACPI_RESOURCE, Resource, Resource->Length);
390     }

392     /* Did not find an EndTag resource descriptor */

394     return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
395 }


398 /*******************************************************************************
399  *
400  * FUNCTION:    AcpiRsGetListLength
401  *
402  * PARAMETERS:  AmlBuffer           - Pointer to the resource byte stream
403  *              AmlBufferLength     - Size of AmlBuffer
404  *              SizeNeeded          - Where the size needed is returned
405  *
406  * RETURN:      Status
407  *
408  * DESCRIPTION: Takes an external resource byte stream and calculates the size
409  *              buffer needed to hold the corresponding internal resource
```



```
410  *              descriptor linked list.
411  *
412  ******************************************************************************/

414 ACPI_STATUS
415 AcpiRsGetListLength (
416     UINT8                   *AmlBuffer,
417     UINT32                  AmlBufferLength,
418     ACPI_SIZE               *SizeNeeded)
419 {
420     ACPI_STATUS             Status;
421     UINT8                   *EndAml;
422     UINT8                   *Buffer;
423     UINT32                  BufferSize;
424     UINT16                  Temp16;
425     UINT16                  ResourceLength;
426     UINT32                  ExtraStructBytes;
427     UINT8                   ResourceIndex;
428     UINT8                   MinimumAmlResourceLength;
429     AML_RESOURCE            *AmlResource;


432     ACPI_FUNCTION_TRACE (RsGetListLength);


435     *SizeNeeded = ACPI_RS_SIZE_MIN;         /* Minimum size is one EndTag */
403     *SizeNeeded = 0;
436     EndAml = AmlBuffer + AmlBufferLength;

438     /* Walk the list of AML resource descriptors */

440     while (AmlBuffer < EndAml)
441     {
442         /* Validate the Resource Type and Resource Length */

444         Status = AcpiUtValidateResource (NULL, AmlBuffer, &ResourceIndex);
412         Status = AcpiUtValidateResource (AmlBuffer, &ResourceIndex);
445         if (ACPI_FAILURE (Status))
446         {
447             /*
448              * Exit on failure. Cannot continue because the descriptor length
449              * may be bogus also.
450              */
451             return_ACPI_STATUS (Status);
452         }

454         AmlResource = (void *) AmlBuffer;

456         /* Get the resource length and base (minimum) AML size */

458         ResourceLength = AcpiUtGetResourceLength (AmlBuffer);
459         MinimumAmlResourceLength = AcpiGbl_ResourceAmlSizes[ResourceIndex];

461         /*
462          * Augment the size for descriptors with optional
463          * and/or variable length fields
464          */
465         ExtraStructBytes = 0;
466         Buffer = AmlBuffer + AcpiUtGetResourceHeaderLength (AmlBuffer);

468         switch (AcpiUtGetResourceType (AmlBuffer))
469         {
470         case ACPI_RESOURCE_NAME_IRQ:
471             /*
472              * IRQ Resource:
473              * Get the number of bits set in the 16-bit IRQ mask
```

```
474              */
475             ACPI_MOVE_16_TO_16 (&Temp16, Buffer);
476             ExtraStructBytes = AcpiRsCountSetBits (Temp16);
477             break;


480         case ACPI_RESOURCE_NAME_DMA:
481             /*
482              * DMA Resource:
483              * Get the number of bits set in the 8-bit DMA mask
484              */
485             ExtraStructBytes = AcpiRsCountSetBits (*Buffer);
486             break;


489         case ACPI_RESOURCE_NAME_VENDOR_SMALL:
490         case ACPI_RESOURCE_NAME_VENDOR_LARGE:
491             /*
492              * Vendor Resource:
493              * Get the number of vendor data bytes
494              */
495             ExtraStructBytes = ResourceLength;

497             /*
498              * There is already one byte included in the minimum
499              * descriptor size. If there are extra struct bytes,
500              * subtract one from the count.
501              */
502             if (ExtraStructBytes)
503             {
504                 ExtraStructBytes--;
505             }
506             break;


509         case ACPI_RESOURCE_NAME_END_TAG:
510             /*
511              * End Tag: This is the normal exit
463              * End Tag:
464              * This is the normal exit, add size of EndTag
512              */
466             *SizeNeeded += ACPI_RS_SIZE_MIN;
513             return_ACPI_STATUS (AE_OK);


516         case ACPI_RESOURCE_NAME_ADDRESS32:
517         case ACPI_RESOURCE_NAME_ADDRESS16:
518         case ACPI_RESOURCE_NAME_ADDRESS64:
519             /*
520              * Address Resource:
521              * Add the size of the optional ResourceSource
522              */
523             ExtraStructBytes = AcpiRsStreamOptionLength (
524                 ResourceLength, MinimumAmlResourceLength);
525             break;


528         case ACPI_RESOURCE_NAME_EXTENDED_IRQ:
529             /*
530              * Extended IRQ Resource:
531              * Using the InterruptTableLength, add 4 bytes for each additional
532              * interrupt. Note: at least one interrupt is required and is
533              * included in the minimum descriptor size (reason for the -1)
534              */
535             ExtraStructBytes = (Buffer[1] - 1) * sizeof (UINT32);
```



```
537             /* Add the size of the optional ResourceSource */

539             ExtraStructBytes += AcpiRsStreamOptionLength (
540                 ResourceLength - ExtraStructBytes, MinimumAmlResourceLength);
541             break;

543         case ACPI_RESOURCE_NAME_GPIO:

545             /* Vendor data is optional */

547             if (AmlResource->Gpio.VendorLength)
548             {
549                 ExtraStructBytes += AmlResource->Gpio.VendorOffset -
550                     AmlResource->Gpio.PinTableOffset + AmlResource->Gpio.VendorL
551             }
552             else
553             {
554                 ExtraStructBytes += AmlResource->LargeHeader.ResourceLength +
555                     sizeof (AML_RESOURCE_LARGE_HEADER) -
556                     AmlResource->Gpio.PinTableOffset;
557             }
558             break;

560         case ACPI_RESOURCE_NAME_SERIAL_BUS:

562             MinimumAmlResourceLength = AcpiGbl_ResourceAmlSerialBusSizes[
563                 AmlResource->CommonSerialBus.Type];
564             ExtraStructBytes += AmlResource->CommonSerialBus.ResourceLength -
565                 MinimumAmlResourceLength;
566             break;

568         default:

570             break;
571         }

573         /*
574          * Update the required buffer size for the internal descriptor structs
575          *
576          * Important: Round the size up for the appropriate alignment. This
577          * is a requirement on IA64.
578          */
579         if (AcpiUtGetResourceType (AmlBuffer) == ACPI_RESOURCE_NAME_SERIAL_BUS)
580         {
581             BufferSize = AcpiGbl_ResourceStructSerialBusSizes[
582                 AmlResource->CommonSerialBus.Type] + ExtraStructBytes;
583         }
584         else
585         {
586             BufferSize = AcpiGbl_ResourceStructSizes[ResourceIndex] +
587                         ExtraStructBytes;
588         }
589         BufferSize = (UINT32) ACPI_ROUND_UP_TO_NATIVE_WORD (BufferSize);

591         *SizeNeeded += BufferSize;

593         ACPI_DEBUG_PRINT ((ACPI_DB_RESOURCES,
594             "Type %.2X, AmlLength %.2X InternalLength %.2X\n",
595             AcpiUtGetResourceType (AmlBuffer),
596             AcpiUtGetDescriptorLength (AmlBuffer), BufferSize));

598         /*
599          * Point to the next resource within the AML stream using the length
600          * contained in the resource descriptor header
601          */
602         AmlBuffer += AcpiUtGetDescriptorLength (AmlBuffer);
```

```
603     }

605     /* Did not find an EndTag resource descriptor */

607     return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
608 }


611 /*******************************************************************************
612  *
613  * FUNCTION:    AcpiRsGetPciRoutingTableLength
614  *
615  * PARAMETERS:  PackageObject           - Pointer to the package object
616  *              BufferSizeNeeded        - UINT32 pointer of the size buffer
617  *                                        needed to properly return the
618  *                                        parsed data
619  *
620  * RETURN:      Status
621  *
622  * DESCRIPTION: Given a package representing a PCI routing table, this
623  *              calculates the size of the corresponding linked list of
624  *              descriptions.
625  *
626  ******************************************************************************/

628 ACPI_STATUS
629 AcpiRsGetPciRoutingTableLength (
630     ACPI_OPERAND_OBJECT     *PackageObject,
631     ACPI_SIZE               *BufferSizeNeeded)
632 {
633     UINT32                  NumberOfElements;
634     ACPI_SIZE               TempSizeNeeded = 0;
635     ACPI_OPERAND_OBJECT     **TopObjectList;
636     UINT32                  Index;
637     ACPI_OPERAND_OBJECT     *PackageElement;
638     ACPI_OPERAND_OBJECT     **SubObjectList;
639     BOOLEAN                 NameFound;
640     UINT32                  TableIndex;


643     ACPI_FUNCTION_TRACE (RsGetPciRoutingTableLength);


646     NumberOfElements = PackageObject->Package.Count;

648     /*
649      * Calculate the size of the return buffer.
650      * The base size is the number of elements * the sizes of the
651      * structures. Additional space for the strings is added below.
652      * The minus one is to subtract the size of the UINT8 Source[1]
653      * member because it is added below.
654      *
655      * But each PRT_ENTRY structure has a pointer to a string and
656      * the size of that string must be found.
657      */
658     TopObjectList = PackageObject->Package.Elements;

660     for (Index = 0; Index < NumberOfElements; Index++)
661     {
662         /* Dereference the sub-package */

664         PackageElement = *TopObjectList;

666         /* We must have a valid Package object */

668         if (!PackageElement ||
```



```
669             (PackageElement->Common.Type != ACPI_TYPE_PACKAGE))
670         {
671             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
672         }

674         /*
675          * The SubObjectList will now point to an array of the
676          * four IRQ elements: Address, Pin, Source and SourceIndex
677          */
678         SubObjectList = PackageElement->Package.Elements;

680         /* Scan the IrqTableElements for the Source Name String */

682         NameFound = FALSE;

684         for (TableIndex = 0;
685              TableIndex < PackageElement->Package.Count && !NameFound;
686              TableIndex++)
605         for (TableIndex = 0; TableIndex < 4 && !NameFound; TableIndex++)
687         {
688             if (*SubObjectList && /* Null object allowed */

690                 ((ACPI_TYPE_STRING ==
691                     (*SubObjectList)->Common.Type) ||

693                 ((ACPI_TYPE_LOCAL_REFERENCE ==
694                     (*SubObjectList)->Common.Type) &&

696                     ((*SubObjectList)->Reference.Class ==
697                         ACPI_REFCLASS_NAME))))
698             {
699                 NameFound = TRUE;
700             }
701             else
702             {
703                 /* Look at the next element */

705                 SubObjectList++;
706             }
707         }

709         TempSizeNeeded += (sizeof (ACPI_PCI_ROUTING_TABLE) - 4);

711         /* Was a String type found? */

713         if (NameFound)
714         {
715             if ((*SubObjectList)->Common.Type == ACPI_TYPE_STRING)
716             {
717                 /*
718                  * The length String.Length field does not include the
719                  * terminating NULL, add 1
720                  */
721                 TempSizeNeeded += ((ACPI_SIZE)
722                     (*SubObjectList)->String.Length + 1);
723             }
724             else
725             {
726                 TempSizeNeeded += AcpiNsGetPathnameLength (
727                                       (*SubObjectList)->Reference.Node);
728             }
729         }
730         else
731         {
732             /*
733              * If no name was found, then this is a NULL, which is
```

```
 734              * translated as a UINT32 zero.
 735              */
 736             TempSizeNeeded += sizeof (UINT32);
 737         }

 739         /* Round up the size since each element must be aligned */

 741         TempSizeNeeded = ACPI_ROUND_UP_TO_64BIT (TempSizeNeeded);

 743         /* Point to the next ACPI_OPERAND_OBJECT */

 745         TopObjectList++;
 746     }

 748     /*
 749      * Add an extra element to the end of the list, essentially a
 750      * NULL terminator
 751      */
 752     *BufferSizeNeeded = TempSizeNeeded + sizeof (ACPI_PCI_ROUTING_TABLE);
 753     return_ACPI_STATUS (AE_OK);
 754 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   16706 Thu Dec 26 13:49:28 2013
new/usr/src/common/acpica/components/resources/rscreate.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rscreate - Create resource lists/tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSCREATE_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"
#include "acnamesp.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rscreate")


/*******************************************************************************
 *
 * FUNCTION:    AcpiBufferToResource
 *
```



```
 * PARAMETERS:  AmlBuffer           - Pointer to the resource byte stream
 *              AmlBufferLength     - Length of the AmlBuffer
 *              ResourcePtr         - Where the converted resource is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert a raw AML buffer to a resource list
 *
 ******************************************************************************/

ACPI_STATUS
AcpiBufferToResource (
    UINT8                   *AmlBuffer,
    UINT16                  AmlBufferLength,
    ACPI_RESOURCE           **ResourcePtr)
{
    ACPI_STATUS             Status;
    ACPI_SIZE               ListSizeNeeded;
    void                    *Resource;
    void                    *CurrentResourcePtr;

    /*
     * Note: we allow AE_AML_NO_RESOURCE_END_TAG, since an end tag
     * is not required here.
     */

    /* Get the required length for the converted resource */

    Status = AcpiRsGetListLength (AmlBuffer, AmlBufferLength,
                &ListSizeNeeded);
    if (Status == AE_AML_NO_RESOURCE_END_TAG)
    {
        Status = AE_OK;
    }
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Allocate a buffer for the converted resource */

    Resource = ACPI_ALLOCATE_ZEROED (ListSizeNeeded);
    CurrentResourcePtr = Resource;
    if (!Resource)
    {
        return (AE_NO_MEMORY);
    }

    /* Perform the AML-to-Resource conversion */

    Status = AcpiUtWalkAmlResources (NULL, AmlBuffer, AmlBufferLength,
                AcpiRsConvertAmlToResources, &CurrentResourcePtr);
    if (Status == AE_AML_NO_RESOURCE_END_TAG)
    {
        Status = AE_OK;
    }
    if (ACPI_FAILURE (Status))
    {
        ACPI_FREE (Resource);
    }
    else
    {
        *ResourcePtr = Resource;
    }

    return (Status);
```

```
125 }


128 /*******************************************************************************
129  *
130  * FUNCTION:    AcpiRsCreateResourceList
131  *
132  * PARAMETERS:  AmlBuffer           - Pointer to the resource byte stream
133  *              OutputBuffer        - Pointer to the user's buffer
134  *
135  * RETURN:      Status: AE_OK if okay, else a valid ACPI_STATUS code
136  *              If OutputBuffer is not large enough, OutputBufferLength
137  *              indicates how large OutputBuffer should be, else it
138  *              indicates how may UINT8 elements of OutputBuffer are valid.
139  *
140  * DESCRIPTION: Takes the byte stream returned from a _CRS, _PRS control method
141  *              execution and parses the stream to create a linked list
142  *              of device resources.
143  *
144  ******************************************************************************/

146 ACPI_STATUS
147 AcpiRsCreateResourceList (
148     ACPI_OPERAND_OBJECT     *AmlBuffer,
149     ACPI_BUFFER             *OutputBuffer)
150 {

152     ACPI_STATUS             Status;
153     UINT8                   *AmlStart;
154     ACPI_SIZE               ListSizeNeeded = 0;
155     UINT32                  AmlBufferLength;
156     void                    *Resource;


159     ACPI_FUNCTION_TRACE (RsCreateResourceList);


162     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "AmlBuffer = %p\n",
163         AmlBuffer));

165     /* Params already validated, so we don't re-validate here */

167     AmlBufferLength = AmlBuffer->Buffer.Length;
168     AmlStart = AmlBuffer->Buffer.Pointer;

170     /*
171      * Pass the AmlBuffer into a module that can calculate
172      * the buffer size needed for the linked list
173      */
174     Status = AcpiRsGetListLength (AmlStart, AmlBufferLength,
175                 &ListSizeNeeded);

177     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Status=%X ListSizeNeeded=%X\n",
178         Status, (UINT32) ListSizeNeeded));
179     if (ACPI_FAILURE (Status))
180     {
181         return_ACPI_STATUS (Status);
182     }

184     /* Validate/Allocate/Clear caller buffer */

186     Status = AcpiUtInitializeBuffer (OutputBuffer, ListSizeNeeded);
187     if (ACPI_FAILURE (Status))
188     {
189         return_ACPI_STATUS (Status);
190     }
```



```
192     /* Do the conversion */

194     Resource = OutputBuffer->Pointer;
195     Status = AcpiUtWalkAmlResources (NULL, AmlStart, AmlBufferLength,
122     Status = AcpiUtWalkAmlResources (AmlStart, AmlBufferLength,
196                 AcpiRsConvertAmlToResources, &Resource);
197     if (ACPI_FAILURE (Status))
198     {
199         return_ACPI_STATUS (Status);
200     }

202     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "OutputBuffer %p Length %X\n",
203             OutputBuffer->Pointer, (UINT32) OutputBuffer->Length));
204     return_ACPI_STATUS (AE_OK);
205 }


208 /*******************************************************************************
209  *
210  * FUNCTION:    AcpiRsCreatePciRoutingTable
211  *
212  * PARAMETERS:  PackageObject           - Pointer to a package containing one
213  *                                        of more ACPI_OPERAND_OBJECTs
139  * PARAMETERS:  PackageObject           - Pointer to an ACPI_OPERAND_OBJECT
140  *                                        package
214  *              OutputBuffer            - Pointer to the user's buffer
215  *
216  * RETURN:      Status  AE_OK if okay, else a valid ACPI_STATUS code.
217  *              If the OutputBuffer is too small, the error will be
218  *              AE_BUFFER_OVERFLOW and OutputBuffer->Length will point
219  *              to the size buffer needed.
220  *
221  * DESCRIPTION: Takes the ACPI_OPERAND_OBJECT package and creates a
222  *              linked list of PCI interrupt descriptions
223  *
224  * NOTE: It is the caller's responsibility to ensure that the start of the
225  * output buffer is aligned properly (if necessary).
226  *
227  ******************************************************************************/

229 ACPI_STATUS
230 AcpiRsCreatePciRoutingTable (
231     ACPI_OPERAND_OBJECT     *PackageObject,
232     ACPI_BUFFER             *OutputBuffer)
233 {
234     UINT8                   *Buffer;
235     ACPI_OPERAND_OBJECT     **TopObjectList;
236     ACPI_OPERAND_OBJECT     **SubObjectList;
237     ACPI_OPERAND_OBJECT     *ObjDesc;
238     ACPI_SIZE               BufferSizeNeeded = 0;
239     UINT32                  NumberOfElements;
240     UINT32                  Index;
241     ACPI_PCI_ROUTING_TABLE  *UserPrt;
242     ACPI_NAMESPACE_NODE     *Node;
243     ACPI_STATUS             Status;
244     ACPI_BUFFER             PathBuffer;


247     ACPI_FUNCTION_TRACE (RsCreatePciRoutingTable);


250     /* Params already validated, so we don't re-validate here */

252     /* Get the required buffer length */
```

```
254     Status = AcpiRsGetPciRoutingTableLength (PackageObject,
255                 &BufferSizeNeeded);
256     if (ACPI_FAILURE (Status))
257     {
258         return_ACPI_STATUS (Status);
259     }

261     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "BufferSizeNeeded = %X\n",
262         (UINT32) BufferSizeNeeded));

264     /* Validate/Allocate/Clear caller buffer */

266     Status = AcpiUtInitializeBuffer (OutputBuffer, BufferSizeNeeded);
267     if (ACPI_FAILURE (Status))
268     {
269         return_ACPI_STATUS (Status);
270     }

272     /*
273      * Loop through the ACPI_INTERNAL_OBJECTS - Each object should be a
274      * package that in turn contains an UINT64 Address, a UINT8 Pin,
275      * a Name, and a UINT8 SourceIndex.
276      */
277     TopObjectList    = PackageObject->Package.Elements;
278     NumberOfElements = PackageObject->Package.Count;
279     Buffer           = OutputBuffer->Pointer;
280     UserPrt          = ACPI_CAST_PTR (ACPI_PCI_ROUTING_TABLE, Buffer);

282     for (Index = 0; Index < NumberOfElements; Index++)
283     {
284         /*
285          * Point UserPrt past this current structure
286          *
287          * NOTE: On the first iteration, UserPrt->Length will
288          * be zero because we cleared the return buffer earlier
289          */
290         Buffer += UserPrt->Length;
291         UserPrt = ACPI_CAST_PTR (ACPI_PCI_ROUTING_TABLE, Buffer);

293         /*
294          * Fill in the Length field with the information we have at this point.
295          * The minus four is to subtract the size of the UINT8 Source[4] member
296          * because it is added below.
297          */
298         UserPrt->Length = (sizeof (ACPI_PCI_ROUTING_TABLE) - 4);

227         /* Each element of the top-level package must also be a package */

229         if ((*TopObjectList)->Common.Type != ACPI_TYPE_PACKAGE)
230         {
231             ACPI_ERROR ((AE_INFO,
232                 "(PRT[%u]) Need sub-package, found %s",
233                 Index, AcpiUtGetObjectTypeName (*TopObjectList)));
234             return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
235         }

300         /* Each sub-package must be of length 4 */

302         if ((*TopObjectList)->Package.Count != 4)
303         {
304             ACPI_ERROR ((AE_INFO,
305                 "(PRT[%u]) Need package of length 4, found length %u",
306                 Index, (*TopObjectList)->Package.Count));
307             return_ACPI_STATUS (AE_AML_PACKAGE_LIMIT);
308         }
```



```
310         /*
311          * Dereference the sub-package.
312          * The SubObjectList will now point to an array of the four IRQ
313          * elements: [Address, Pin, Source, SourceIndex]
314          */
315         SubObjectList = (*TopObjectList)->Package.Elements;

317         /* 1) First subobject: Dereference the PRT.Address */

319         ObjDesc = SubObjectList[0];
320         if (ObjDesc->Common.Type != ACPI_TYPE_INTEGER)
321         {
322             ACPI_ERROR ((AE_INFO, "(PRT[%u].Address) Need Integer, found %s",
323                 Index, AcpiUtGetObjectTypeName (ObjDesc)));
324             return_ACPI_STATUS (AE_BAD_DATA);
325         }

327         UserPrt->Address = ObjDesc->Integer.Value;

329         /* 2) Second subobject: Dereference the PRT.Pin */

331         ObjDesc = SubObjectList[1];
332         if (ObjDesc->Common.Type != ACPI_TYPE_INTEGER)
333         {
334             ACPI_ERROR ((AE_INFO, "(PRT[%u].Pin) Need Integer, found %s",
335                 Index, AcpiUtGetObjectTypeName (ObjDesc)));
336             return_ACPI_STATUS (AE_BAD_DATA);
337         }

339         UserPrt->Pin = (UINT32) ObjDesc->Integer.Value;

341         /*
279          * If the BIOS has erroneously reversed the _PRT SourceName (index 2)
280          * and the SourceIndex (index 3), fix it. _PRT is important enough to
281          * workaround this BIOS error. This also provides compatibility with
282          * other ACPI implementations.
283          */
284         ObjDesc = SubObjectList[3];
285         if (!ObjDesc || (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
286         {
287             SubObjectList[3] = SubObjectList[2];
288             SubObjectList[2] = ObjDesc;

290             ACPI_WARNING ((AE_INFO,
291                 "(PRT[%X].Source) SourceName and SourceIndex are reversed, fixed
292                 Index));
293         }

295         /*
342          * 3) Third subobject: Dereference the PRT.SourceName
343          * The name may be unresolved (slack mode), so allow a null object
344          */
345         ObjDesc = SubObjectList[2];
346         if (ObjDesc)
347         {
348             switch (ObjDesc->Common.Type)
349             {
350             case ACPI_TYPE_LOCAL_REFERENCE:

352                 if (ObjDesc->Reference.Class != ACPI_REFCLASS_NAME)
353                 {
354                     ACPI_ERROR ((AE_INFO,
355                         "(PRT[%u].Source) Need name, found Reference Class 0x%X"
356                         Index, ObjDesc->Reference.Class));
357                     return_ACPI_STATUS (AE_BAD_DATA);
358                 }
```

```
360                 Node = ObjDesc->Reference.Node;

362                 /* Use *remaining* length of the buffer as max for pathname */

364                 PathBuffer.Length = OutputBuffer->Length -
365                                     (UINT32) ((UINT8 *) UserPrt->Source -
366                                     (UINT8 *) OutputBuffer->Pointer);
367                 PathBuffer.Pointer = UserPrt->Source;

369                 Status = AcpiNsHandleToPathname ((ACPI_HANDLE) Node, &PathBuffer

371                 /* +1 to include null terminator */

373                 UserPrt->Length += (UINT32) ACPI_STRLEN (UserPrt->Source) + 1;
374                 break;


376             case ACPI_TYPE_STRING:

378                 ACPI_STRCPY (UserPrt->Source, ObjDesc->String.Pointer);

380                 /*
381                  * Add to the Length field the length of the string
382                  * (add 1 for terminator)
383                  */
384                 UserPrt->Length += ObjDesc->String.Length + 1;
385                 break;


387             case ACPI_TYPE_INTEGER:
388                 /*
389                  * If this is a number, then the Source Name is NULL, since the
390                  * entire buffer was zeroed out, we can leave this alone.
391                  *
392                  * Add to the Length field the length of the UINT32 NULL
393                  */
394                 UserPrt->Length += sizeof (UINT32);
395                 break;


397             default:

399                ACPI_ERROR ((AE_INFO,
400                    "(PRT[%u].Source) Need Ref/String/Integer, found %s",
401                    Index, AcpiUtGetObjectTypeName (ObjDesc)));
402                return_ACPI_STATUS (AE_BAD_DATA);
403             }
404         }

406         /* Now align the current length */

408         UserPrt->Length = (UINT32) ACPI_ROUND_UP_TO_64BIT (UserPrt->Length);

410         /* 4) Fourth subobject: Dereference the PRT.SourceIndex */

412         ObjDesc = SubObjectList[3];
413         if (ObjDesc->Common.Type != ACPI_TYPE_INTEGER)
414         {
415             ACPI_ERROR ((AE_INFO,
416                 "(PRT[%u].SourceIndex) Need Integer, found %s",
417                 Index, AcpiUtGetObjectTypeName (ObjDesc)));
418             return_ACPI_STATUS (AE_BAD_DATA);
419         }

421         UserPrt->SourceIndex = (UINT32) ObjDesc->Integer.Value;
```



```
423         /* Point to the next ACPI_OPERAND_OBJECT in the top level package */

425         TopObjectList++;
426     }

428     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "OutputBuffer %p Length %X\n",
429             OutputBuffer->Pointer, (UINT32) OutputBuffer->Length));
430     return_ACPI_STATUS (AE_OK);
431 }


434 /*******************************************************************************
435  *
436  * FUNCTION:    AcpiRsCreateAmlResources
437  *
438  * PARAMETERS:  ResourceList            - Pointer to the resource list buffer
439  *              OutputBuffer            - Where the AML buffer is returned
395  * PARAMETERS:  LinkedListBuffer        - Pointer to the resource linked list
396  *              OutputBuffer            - Pointer to the user's buffer
440  *
441  * RETURN:      Status  AE_OK if okay, else a valid ACPI_STATUS code.
442  *              If the OutputBuffer is too small, the error will be
443  *              AE_BUFFER_OVERFLOW and OutputBuffer->Length will point
444  *              to the size buffer needed.
445  *
446  * DESCRIPTION: Converts a list of device resources to an AML bytestream
447  *              to be used as input for the _SRS control method.
403  * DESCRIPTION: Takes the linked list of device resources and
404  *              creates a bytestream to be used as input for the
405  *              _SRS control method.
448  *
449  ******************************************************************************/

451 ACPI_STATUS
452 AcpiRsCreateAmlResources (
453     ACPI_BUFFER             *ResourceList,
411     ACPI_RESOURCE           *LinkedListBuffer,
454     ACPI_BUFFER             *OutputBuffer)
455 {
456     ACPI_STATUS             Status;
457     ACPI_SIZE               AmlSizeNeeded = 0;


460     ACPI_FUNCTION_TRACE (RsCreateAmlResources);


463     /* Params already validated, no need to re-validate here */
421     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "LinkedListBuffer = %p\n",
422         LinkedListBuffer));

465     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "ResourceList Buffer = %p\n",
466         ResourceList->Pointer));
424     /*
425      * Params already validated, so we don't re-validate here
426      *
427      * Pass the LinkedListBuffer into a module that calculates
428      * the buffer size needed for the byte stream.
429      */
430     Status = AcpiRsGetAmlLength (LinkedListBuffer,
431                 &AmlSizeNeeded);

468     /* Get the buffer size needed for the AML byte stream */

470     Status = AcpiRsGetAmlLength (ResourceList->Pointer,
471                 ResourceList->Length, &AmlSizeNeeded);
```

```
473     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "AmlSizeNeeded=%X, %s\n",
474         (UINT32) AmlSizeNeeded, AcpiFormatException (Status)));
475     if (ACPI_FAILURE (Status))
476     {
477         return_ACPI_STATUS (Status);
478     }

480     /* Validate/Allocate/Clear caller buffer */

482     Status = AcpiUtInitializeBuffer (OutputBuffer, AmlSizeNeeded);
483     if (ACPI_FAILURE (Status))
484     {
485         return_ACPI_STATUS (Status);
486     }

488     /* Do the conversion */

490     Status = AcpiRsConvertResourcesToAml (ResourceList->Pointer,
491                 AmlSizeNeeded, OutputBuffer->Pointer);
450     Status = AcpiRsConvertResourcesToAml (LinkedListBuffer, AmlSizeNeeded,
451                     OutputBuffer->Pointer);
492     if (ACPI_FAILURE (Status))
493     {
494         return_ACPI_STATUS (Status);
495     }

497     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "OutputBuffer %p Length %X\n",
498         OutputBuffer->Pointer, (UINT32) OutputBuffer->Length));
499     return_ACPI_STATUS (AE_OK);
500 }
```

```
**********************************************************
   17150 Thu Dec 26 13:49:29 2013
new/usr/src/common/acpica/components/resources/rsdump.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsdump - Functions to display the resource structures.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __RSDUMP_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsdump")


#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

/* Local prototypes */

static void
AcpiRsOutString (
```



```
    char                    *Title,
    char                    *Value);

static void
AcpiRsOutInteger8 (
    char                    *Title,
    UINT8                   Value);

static void
AcpiRsOutInteger16 (
    char                    *Title,
    UINT16                  Value);

static void
AcpiRsOutInteger32 (
    char                    *Title,
    UINT32                  Value);

static void
AcpiRsOutInteger64 (
    char                    *Title,
    UINT64                  Value);

static void
AcpiRsOutTitle (
    char                    *Title);

static void
AcpiRsDumpByteList (
    UINT16                  Length,
    UINT8                   *Data);

static void
AcpiRsDumpWordList (
    UINT16                   Length,
    UINT16                   *Data);

static void
AcpiRsDumpDwordList (
    UINT8                   Length,
    UINT32                  *Data);

static void
AcpiRsDumpShortByteList (
    UINT8                  Length,
    UINT8                  *Data);

static void
AcpiRsDumpResourceSource (
    ACPI_RESOURCE_SOURCE    *ResourceSource);

static void
AcpiRsDumpAddressCommon (
    ACPI_RESOURCE_DATA      *Resource);

static void
AcpiRsDumpDescriptor (
    void                    *Resource,
    ACPI_RSDUMP_INFO *Table);


/*******************************************************************************
 *
 * FUNCTION:    AcpiRsDumpDescriptor
 *
 * PARAMETERS:  Resource            - Buffer containing the resource
```

```
 *              Table               - Table entry to decode the resource
 *
 * RETURN:      None
 *
 * DESCRIPTION: Dump a resource descriptor based on a dump table entry.
 *
 ******************************************************************************/

static void
AcpiRsDumpDescriptor (
    void                    *Resource,
    ACPI_RSDUMP_INFO        *Table)
{
    UINT8                   *Target = NULL;
    UINT8                   *PreviousTarget;
    char                    *Name;
    UINT8                    Count;


    /* First table entry must contain the table length (# of table entries) */

    Count = Table->Offset;

    while (Count)
    {
        PreviousTarget = Target;
        Target = ACPI_ADD_PTR (UINT8, Resource, Table->Offset);
        Name = Table->Name;

        switch (Table->Opcode)
        {
        case ACPI_RSD_TITLE:
            /*
             * Optional resource title
             */
            if (Table->Name)
            {
                AcpiOsPrintf ("%s Resource\n", Name);
            }
            break;

        /* Strings */

        case ACPI_RSD_LITERAL:

            AcpiRsOutString (Name, ACPI_CAST_PTR (char, Table->Pointer));
            break;

        case ACPI_RSD_STRING:

            AcpiRsOutString (Name, ACPI_CAST_PTR (char, Target));
            break;

        /* Data items, 8/16/32/64 bit */

        case ACPI_RSD_UINT8:

            if (Table->Pointer)
            {
                AcpiRsOutString (Name, ACPI_CAST_PTR (char,
                    Table->Pointer [*Target]));
            }
            else
            {
                AcpiRsOutInteger8 (Name, ACPI_GET8 (Target));
            }
```



```
            break;

        case ACPI_RSD_UINT16:

            AcpiRsOutInteger16 (Name, ACPI_GET16 (Target));
            break;

        case ACPI_RSD_UINT32:

            AcpiRsOutInteger32 (Name, ACPI_GET32 (Target));
            break;

        case ACPI_RSD_UINT64:

            AcpiRsOutInteger64 (Name, ACPI_GET64 (Target));
            break;

        /* Flags: 1-bit and 2-bit flags supported */

        case ACPI_RSD_1BITFLAG:

            AcpiRsOutString (Name, ACPI_CAST_PTR (char,
                Table->Pointer [*Target & 0x01]));
            break;

        case ACPI_RSD_2BITFLAG:

            AcpiRsOutString (Name, ACPI_CAST_PTR (char,
                Table->Pointer [*Target & 0x03]));
            break;

        case ACPI_RSD_3BITFLAG:

            AcpiRsOutString (Name, ACPI_CAST_PTR (char,
                Table->Pointer [*Target & 0x07]));
            break;

        case ACPI_RSD_SHORTLIST:
            /*
             * Short byte list (single line output) for DMA and IRQ resources
             * Note: The list length is obtained from the previous table entry
             */
            if (PreviousTarget)
            {
                AcpiRsOutTitle (Name);
                AcpiRsDumpShortByteList (*PreviousTarget, Target);
            }
            break;

        case ACPI_RSD_SHORTLISTX:
            /*
             * Short byte list (single line output) for GPIO vendor data
             * Note: The list length is obtained from the previous table entry
             */
            if (PreviousTarget)
            {
                AcpiRsOutTitle (Name);
                AcpiRsDumpShortByteList (*PreviousTarget,
                    *(ACPI_CAST_INDIRECT_PTR (UINT8, Target)));
            }
            break;

        case ACPI_RSD_LONGLIST:
            /*
             * Long byte list for Vendor resource data
             * Note: The list length is obtained from the previous table entry
```

```
259              */
260             if (PreviousTarget)
261             {
262                 AcpiRsDumpByteList (ACPI_GET16 (PreviousTarget), Target);
263             }
264             break;

266         case ACPI_RSD_DWORDLIST:
267             /*
268              * Dword list for Extended Interrupt resources
269              * Note: The list length is obtained from the previous table entry
270              */
271             if (PreviousTarget)
272             {
273                 AcpiRsDumpDwordList (*PreviousTarget,
274                     ACPI_CAST_PTR (UINT32, Target));
275             }
276             break;

278         case ACPI_RSD_WORDLIST:
279             /*
280              * Word list for GPIO Pin Table
281              * Note: The list length is obtained from the previous table entry
282              */
283             if (PreviousTarget)
284             {
285                 AcpiRsDumpWordList (*PreviousTarget,
286                     *(ACPI_CAST_INDIRECT_PTR (UINT16, Target)));
287             }
288             break;

290         case ACPI_RSD_ADDRESS:
291             /*
292              * Common flags for all Address resources
293              */
294             AcpiRsDumpAddressCommon (ACPI_CAST_PTR (ACPI_RESOURCE_DATA, Target))
295             break;

297         case ACPI_RSD_SOURCE:
298             /*
299              * Optional ResourceSource for Address resources
300              */
301             AcpiRsDumpResourceSource (ACPI_CAST_PTR (ACPI_RESOURCE_SOURCE, Targe
302             break;

304         default:

306             AcpiOsPrintf ("**** Invalid table opcode [%X] ****\n",
307                 Table->Opcode);
308             return;
309         }

311         Table++;
312         Count--;
313     }
314 }


317 /*******************************************************************************
318  *
319  * FUNCTION:    AcpiRsDumpResourceSource
320  *
321  * PARAMETERS:  ResourceSource      - Pointer to a Resource Source struct
322  *
323  * RETURN:      None
324  *
```



```
325  * DESCRIPTION: Common routine for dumping the optional ResourceSource and the
326  *              corresponding ResourceSourceIndex.
327  *
328  ******************************************************************************/

330 static void
331 AcpiRsDumpResourceSource (
332     ACPI_RESOURCE_SOURCE    *ResourceSource)
333 {
334     ACPI_FUNCTION_ENTRY ();


337     if (ResourceSource->Index == 0xFF)
338     {
339         return;
340     }

342     AcpiRsOutInteger8 ("Resource Source Index",
343         ResourceSource->Index);

345     AcpiRsOutString ("Resource Source",
346         ResourceSource->StringPtr ?
347             ResourceSource->StringPtr : "[Not Specified]");
348 }


351 /*******************************************************************************
352  *
353  * FUNCTION:    AcpiRsDumpAddressCommon
354  *
355  * PARAMETERS:  Resource        - Pointer to an internal resource descriptor
356  *
357  * RETURN:      None
358  *
359  * DESCRIPTION: Dump the fields that are common to all Address resource
360  *              descriptors
361  *
362  ******************************************************************************/

364 static void
365 AcpiRsDumpAddressCommon (
366     ACPI_RESOURCE_DATA      *Resource)
367 {
368     ACPI_FUNCTION_ENTRY ();


371    /* Decode the type-specific flags */

373     switch (Resource->Address.ResourceType)
374     {
375     case ACPI_MEMORY_RANGE:

377         AcpiRsDumpDescriptor (Resource, AcpiRsDumpMemoryFlags);
378         break;

380     case ACPI_IO_RANGE:

382         AcpiRsDumpDescriptor (Resource, AcpiRsDumpIoFlags);
383         break;

385     case ACPI_BUS_NUMBER_RANGE:

387         AcpiRsOutString ("Resource Type", "Bus Number Range");
388         break;

390     default:
```

```
392         AcpiRsOutInteger8 ("Resource Type",
393             (UINT8) Resource->Address.ResourceType);
394         break;
395     }

397     /* Decode the general flags */

399     AcpiRsDumpDescriptor (Resource, AcpiRsDumpGeneralFlags);
400 }


403 /*******************************************************************************
404  *
405  * FUNCTION:    AcpiRsDumpResourceList
406  *
407  * PARAMETERS:  ResourceList        - Pointer to a resource descriptor list
408  *
409  * RETURN:      None
410  *
411  * DESCRIPTION: Dispatches the structure to the correct dump routine.
412  *
413  ******************************************************************************/

415 void
416 AcpiRsDumpResourceList (
417     ACPI_RESOURCE           *ResourceList)
418 {
419     UINT32                  Count = 0;
420     UINT32                  Type;


423     ACPI_FUNCTION_ENTRY ();


426     /* Check if debug output enabled */

428     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_RESOURCES, _COMPONENT))
429     {
430         return;
431     }

433     /* Walk list and dump all resource descriptors (END_TAG terminates) */

435     do
436     {
437         AcpiOsPrintf ("\n[%02X] ", Count);
438         Count++;

440         /* Validate Type before dispatch */

442         Type = ResourceList->Type;
443         if (Type > ACPI_RESOURCE_TYPE_MAX)
444         {
445             AcpiOsPrintf (
446                 "Invalid descriptor type (%X) in resource list\n",
447                 ResourceList->Type);
448             return;
449         }

451         /* Sanity check the length. It must not be zero, or we loop forever */

453         if (!ResourceList->Length)
454         {
455             AcpiOsPrintf (
456                 "Invalid zero length descriptor in resource list\n");
```



```
457             return;
458         }

460         /* Dump the resource descriptor */

462         if (Type == ACPI_RESOURCE_TYPE_SERIAL_BUS)
463         {
464             AcpiRsDumpDescriptor (&ResourceList->Data,
465                 AcpiGbl_DumpSerialBusDispatch[ResourceList->Data.CommonSerialBus
466         }
467         else
468         {
469             AcpiRsDumpDescriptor (&ResourceList->Data,
470                 AcpiGbl_DumpResourceDispatch[Type]);
471         }

473         /* Point to the next resource structure */

475         ResourceList = ACPI_NEXT_RESOURCE (ResourceList);

477         /* Exit when END_TAG descriptor is reached */

479     } while (Type != ACPI_RESOURCE_TYPE_END_TAG);
480 }


483 /*******************************************************************************
484  *
485  * FUNCTION:    AcpiRsDumpIrqList
486  *
487  * PARAMETERS:  RouteTable      - Pointer to the routing table to dump.
488  *
489  * RETURN:      None
490  *
491  * DESCRIPTION: Print IRQ routing table
492  *
493  ******************************************************************************/

495 void
496 AcpiRsDumpIrqList (
497     UINT8                   *RouteTable)
498 {
499     ACPI_PCI_ROUTING_TABLE  *PrtElement;
500     UINT8                   Count;


503     ACPI_FUNCTION_ENTRY ();


506     /* Check if debug output enabled */

508     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_RESOURCES, _COMPONENT))
509     {
510         return;
511     }

513     PrtElement = ACPI_CAST_PTR (ACPI_PCI_ROUTING_TABLE, RouteTable);

515     /* Dump all table elements, Exit on zero length element */

517     for (Count = 0; PrtElement->Length; Count++)
518     {
519         AcpiOsPrintf ("\n[%02X] PCI IRQ Routing Table Package\n", Count);
520         AcpiRsDumpDescriptor (PrtElement, AcpiRsDumpPrt);

522         PrtElement = ACPI_ADD_PTR (ACPI_PCI_ROUTING_TABLE,
```

```
                        PrtElement, PrtElement->Length);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiRsOut*
 *
 * PARAMETERS:  Title       - Name of the resource field
 *              Value       - Value of the resource field
 *
 * RETURN:      None
 *
 * DESCRIPTION: Miscellaneous helper functions to consistently format the
 *              output of the resource dump routines
 *
 ******************************************************************************/

static void
AcpiRsOutString (
    char                    *Title,
    char                    *Value)
{
    AcpiOsPrintf ("%27s : %s", Title, Value);
    if (!*Value)
    {
        AcpiOsPrintf ("[NULL NAMESTRING]");
    }
    AcpiOsPrintf ("\n");
}

static void
AcpiRsOutInteger8 (
    char                    *Title,
    UINT8                   Value)
{
    AcpiOsPrintf ("%27s : %2.2X\n", Title, Value);
}

static void
AcpiRsOutInteger16 (
    char                    *Title,
    UINT16                  Value)
{
    AcpiOsPrintf ("%27s : %4.4X\n", Title, Value);
}

static void
AcpiRsOutInteger32 (
    char                    *Title,
    UINT32                  Value)
{
    AcpiOsPrintf ("%27s : %8.8X\n", Title, Value);
}

static void
AcpiRsOutInteger64 (
    char                    *Title,
    UINT64                  Value)
{
    AcpiOsPrintf ("%27s : %8.8X%8.8X\n", Title,
        ACPI_FORMAT_UINT64 (Value));
}

static void
```



```
AcpiRsOutTitle (
    char                    *Title)
{
    AcpiOsPrintf ("%27s : ", Title);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiRsDump*List
 *
 * PARAMETERS:  Length      - Number of elements in the list
 *              Data        - Start of the list
 *
 * RETURN:      None
 *
 * DESCRIPTION: Miscellaneous functions to dump lists of raw data
 *
 ******************************************************************************/

static void
AcpiRsDumpByteList (
    UINT16                  Length,
    UINT8                   *Data)
{
    UINT8                   i;


    for (i = 0; i < Length; i++)
    {
        AcpiOsPrintf ("%25s%2.2X : %2.2X\n",
            "Byte", i, Data[i]);
    }
}

static void
AcpiRsDumpShortByteList (
    UINT8                   Length,
    UINT8                   *Data)
{
    UINT8                   i;


    for (i = 0; i < Length; i++)
    {
        AcpiOsPrintf ("%X ", Data[i]);
    }
    AcpiOsPrintf ("\n");
}

static void
AcpiRsDumpDwordList (
    UINT8                   Length,
    UINT32                  *Data)
{
    UINT8                   i;


    for (i = 0; i < Length; i++)
    {
        AcpiOsPrintf ("%25s%2.2X : %8.8X\n",
            "Dword", i, Data[i]);
    }
}

static void
```

```
655 AcpiRsDumpWordList (
656     UINT16                  Length,
657     UINT16                  *Data)
658 {
659     UINT16                  i;


662     for (i = 0; i < Length; i++)
663     {
664         AcpiOsPrintf ("%25s%2.2X : %4.4X\n",
665             "Word", i, Data[i]);
666     }
667 }

669 #endif
```

```
**********************************************************
   23563 Thu Dec 26 13:49:29 2013
new/usr/src/common/acpica/components/resources/rsdumpinfo.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsdumpinfo - Tables used to display resource descriptors.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __RSDUMPINFO_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsdumpinfo")


#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)


#define ACPI_RSD_OFFSET(f)          (UINT8) ACPI_OFFSET (ACPI_RESOURCE_DATA,f)
#define ACPI_PRT_OFFSET(f)          (UINT8) ACPI_OFFSET (ACPI_PCI_ROUTING_TABLE,
#define ACPI_RSD_TABLE_SIZE(name)   (sizeof(name) / sizeof (ACPI_RSDUMP_INFO))
```



```
/******************************************************************************
 *
 * Resource Descriptor info tables
 *
 * Note: The first table entry must be a Title or Literal and must contain
 * the table length (number of table entries)
 *
 *****************************************************************************/

ACPI_RSDUMP_INFO        AcpiRsDumpIrq[7] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpIrq),               "IRQ
    {ACPI_RSD_UINT8 ,   ACPI_RSD_OFFSET (Irq.DescriptorLength),            "Des
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Irq.Triggering),                  "Tri
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Irq.Polarity),                    "Pol
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Irq.Sharable),                    "Sha
    {ACPI_RSD_UINT8 ,   ACPI_RSD_OFFSET (Irq.InterruptCount),              "Int
    {ACPI_RSD_SHORTLIST,ACPI_RSD_OFFSET (Irq.Interrupts[0]),               "Int
};

ACPI_RSDUMP_INFO        AcpiRsDumpDma[6] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpDma),               "DMA
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Dma.Type),                        "Spe
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Dma.BusMaster),                   "Mas
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Dma.Transfer),                    "Tra
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Dma.ChannelCount),                "Cha
    {ACPI_RSD_SHORTLIST,ACPI_RSD_OFFSET (Dma.Channels[0]),                 "Cha
};

ACPI_RSDUMP_INFO        AcpiRsDumpStartDpf[4] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpStartDpf),          "Sta
    {ACPI_RSD_UINT8 ,   ACPI_RSD_OFFSET (StartDpf.DescriptorLength),       "Des
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (StartDpf.CompatibilityPriority),  "Com
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (StartDpf.PerformanceRobustness),  "Per
};

ACPI_RSDUMP_INFO        AcpiRsDumpEndDpf[1] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpEndDpf),            "End
};

ACPI_RSDUMP_INFO        AcpiRsDumpIo[6] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpIo),                "I/O
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Io.IoDecode),                     "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Io.Minimum),                      "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Io.Maximum),                      "Add
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Io.Alignment),                    "Ali
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Io.AddressLength),                "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpFixedIo[3] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedIo),           "Fix
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (FixedIo.Address),                 "Add
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (FixedIo.AddressLength),           "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpVendor[3] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpVendor),            "Ven
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Vendor.ByteLength),               "Len
    {ACPI_RSD_LONGLIST, ACPI_RSD_OFFSET (Vendor.ByteData[0]),              "Ven
```

```
};

ACPI_RSDUMP_INFO        AcpiRsDumpEndTag[1] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpEndTag),             "End
};

ACPI_RSDUMP_INFO        AcpiRsDumpMemory24[6] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemory24),           "24-
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Memory24.WriteProtect),            "Wri
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Memory24.Minimum),                 "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Memory24.Maximum),                 "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Memory24.Alignment),               "Ali
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Memory24.AddressLength),           "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpMemory32[6] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemory32),           "32-
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Memory32.WriteProtect),            "Wri
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Memory32.Minimum),                 "Add
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Memory32.Maximum),                 "Add
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Memory32.Alignment),               "Ali
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Memory32.AddressLength),           "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpFixedMemory32[4] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedMemory32),      "32-
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (FixedMemory32.WriteProtect),       "Wri
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (FixedMemory32.Address),            "Add
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (FixedMemory32.AddressLength),      "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpAddress16[8] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress16),          "16-
    {ACPI_RSD_ADDRESS,  0,                                                  NULL
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Address16.Granularity),            "Gra
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Address16.Minimum),                "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Address16.Maximum),                "Add
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Address16.TranslationOffset),      "Tra
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Address16.AddressLength),          "Add
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (Address16.ResourceSource),         NULL
};

ACPI_RSDUMP_INFO        AcpiRsDumpAddress32[8] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress32),         "32-B
    {ACPI_RSD_ADDRESS,  0,                                                  NULL
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Address32.Granularity),            "Gra
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Address32.Minimum),                "Add
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Address32.Maximum),                "Add
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Address32.TranslationOffset),      "Tra
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (Address32.AddressLength),          "Add
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (Address32.ResourceSource),         NULL
};

ACPI_RSDUMP_INFO        AcpiRsDumpAddress64[8] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress64),          "64-
    {ACPI_RSD_ADDRESS,  0,                                                  NULL
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (Address64.Granularity),            "Gra
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (Address64.Minimum),                "Add
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (Address64.Maximum),                "Add
```



```
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (Address64.TranslationOffset),      "Tra
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (Address64.AddressLength),          "Add
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (Address64.ResourceSource),         NULL
};

ACPI_RSDUMP_INFO        AcpiRsDumpExtAddress64[8] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpExtAddress64),       "64-
    {ACPI_RSD_ADDRESS,  0,                                                  NULL
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.Granularity),         "Gra
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.Minimum),             "Add
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.Maximum),             "Add
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.TranslationOffset),   "Tra
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.AddressLength),       "Add
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (ExtAddress64.TypeSpecific),        "Typ
};

ACPI_RSDUMP_INFO        AcpiRsDumpExtIrq[8] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpExtIrq),             "Ext
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (ExtendedIrq.ProducerConsumer),     "Typ
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (ExtendedIrq.Triggering),           "Tri
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (ExtendedIrq.Polarity),             "Pol
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (ExtendedIrq.Sharable),             "Sha
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (ExtendedIrq.ResourceSource),       NULL
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (ExtendedIrq.InterruptCount),       "Int
    {ACPI_RSD_DWORDLIST,ACPI_RSD_OFFSET (ExtendedIrq.Interrupts[0]),        "Int
};

ACPI_RSDUMP_INFO        AcpiRsDumpGenericReg[6] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpGenericReg),         "Gen
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (GenericReg.SpaceId),               "Spa
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (GenericReg.BitWidth),              "Bit
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (GenericReg.BitOffset),             "Bit
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (GenericReg.AccessSize),            "Acc
    {ACPI_RSD_UINT64,   ACPI_RSD_OFFSET (GenericReg.Address),               "Add
};

ACPI_RSDUMP_INFO        AcpiRsDumpGpio[16] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpGpio),               "GPI
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Gpio.RevisionId),                  "Rev
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Gpio.ConnectionType),              "Con
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Gpio.ProducerConsumer),            "Pro
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (Gpio.PinConfig),                   "Pin
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Gpio.Sharable),                    "Sha
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Gpio.IoRestriction),               "IoR
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Gpio.Triggering),                  "Tri
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Gpio.Polarity),                    "Pol
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Gpio.DriveStrength),               "Dri
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Gpio.DebounceTimeout),             "Deb
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (Gpio.ResourceSource),              "Res
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Gpio.PinTableLength),              "Pin
    {ACPI_RSD_WORDLIST, ACPI_RSD_OFFSET (Gpio.PinTable),                    "Pin
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (Gpio.VendorLength),                "Ven
    {ACPI_RSD_SHORTLISTX,ACPI_RSD_OFFSET (Gpio.VendorData),                 "Ven
};

ACPI_RSDUMP_INFO        AcpiRsDumpFixedDma[4] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedDma),           "Fix
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (FixedDma.RequestLines),            "Req
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (FixedDma.Channels),                "Cha
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (FixedDma.Width),                   "Tra
};
```

```
#define ACPI_RS_DUMP_COMMON_SERIAL_BUS \
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (CommonSerialBus.RevisionId),       "Rev
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (CommonSerialBus.Type),             "Typ
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (CommonSerialBus.ProducerConsumer), "Pro
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (CommonSerialBus.SlaveMode),        "Sla
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (CommonSerialBus.TypeRevisionId),   "Typ
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (CommonSerialBus.TypeDataLength),   "Typ
    {ACPI_RSD_SOURCE,   ACPI_RSD_OFFSET (CommonSerialBus.ResourceSource),   "Res
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (CommonSerialBus.VendorLength),     "Ven
    {ACPI_RSD_SHORTLISTX,ACPI_RSD_OFFSET (CommonSerialBus.VendorData),      "Ven

ACPI_RSDUMP_INFO        AcpiRsDumpCommonSerialBus[10] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpCommonSerialBus),    "Com
    ACPI_RS_DUMP_COMMON_SERIAL_BUS
};

ACPI_RSDUMP_INFO        AcpiRsDumpI2cSerialBus[13] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpI2cSerialBus),       "I2C
    ACPI_RS_DUMP_COMMON_SERIAL_BUS
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (I2cSerialBus.AccessMode),          "Acc
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (I2cSerialBus.ConnectionSpeed),     "Con
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (I2cSerialBus.SlaveAddress),        "Sla
};

ACPI_RSDUMP_INFO        AcpiRsDumpSpiSerialBus[17] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpSpiSerialBus),       "Spi
    ACPI_RS_DUMP_COMMON_SERIAL_BUS
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (SpiSerialBus.WireMode),            "Wir
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (SpiSerialBus.DevicePolarity),      "Dev
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (SpiSerialBus.DataBitLength),       "Dat
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (SpiSerialBus.ClockPhase),          "Clo
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (SpiSerialBus.ClockPolarity),       "Clo
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (SpiSerialBus.DeviceSelection),     "Dev
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (SpiSerialBus.ConnectionSpeed),     "Con
};

ACPI_RSDUMP_INFO        AcpiRsDumpUartSerialBus[19] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpUartSerialBus),       "Ua
    ACPI_RS_DUMP_COMMON_SERIAL_BUS
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.FlowControl),         "Fl
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.StopBits),           "St
    {ACPI_RSD_3BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.DataBits),           "Da
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.Endian),             "En
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (UartSerialBus.Parity),             "Pa
    {ACPI_RSD_UINT8,    ACPI_RSD_OFFSET (UartSerialBus.LinesEnabled),       "Li
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (UartSerialBus.RxFifoSize),         "Rx
    {ACPI_RSD_UINT16,   ACPI_RSD_OFFSET (UartSerialBus.TxFifoSize),         "Tx
    {ACPI_RSD_UINT32,   ACPI_RSD_OFFSET (UartSerialBus.DefaultBaudRate),    "Co
};

/*
 * Tables used for common address descriptor flag fields
 */
ACPI_RSDUMP_INFO        AcpiRsDumpGeneralFlags[5] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpGeneralFlags),       NULL
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.ProducerConsumer),         "Con
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Decode),                   "Add
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.MinAddressFixed),          "Min
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.MaxAddressFixed),          "Max
};
```



```
ACPI_RSDUMP_INFO        AcpiRsDumpMemoryFlags[5] =
{
    {ACPI_RSD_LITERAL,  ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemoryFlags),        "Res
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.WriteProtect),    "Wri
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.Caching),         "Cac
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.RangeType),       "Ran
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.Translation),     "Tra
};

ACPI_RSDUMP_INFO        AcpiRsDumpIoFlags[4] =
{
    {ACPI_RSD_LITERAL,  ACPI_RSD_TABLE_SIZE (AcpiRsDumpIoFlags),            "Res
    {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.RangeType),        "Ran
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.Translation),      "Tra
    {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.TranslationType),  "Tra
};


/*
 * Table used to dump _PRT contents
 */
ACPI_RSDUMP_INFO        AcpiRsDumpPrt[5] =
{
    {ACPI_RSD_TITLE,    ACPI_RSD_TABLE_SIZE (AcpiRsDumpPrt),                NULL
    {ACPI_RSD_UINT64,   ACPI_PRT_OFFSET (Address),                          "Add
    {ACPI_RSD_UINT32,   ACPI_PRT_OFFSET (Pin),                              "Pin
    {ACPI_RSD_STRING,   ACPI_PRT_OFFSET (Source[0]),                        "Sou
    {ACPI_RSD_UINT32,   ACPI_PRT_OFFSET (SourceIndex),                      "Sou
};

#endif
```

****
12406 Thu Dec 26 13:49:29 2013
new/usr/src/common/acpica/components/resources/rsinfo.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: rsinfo - Dispatch and Info tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSINFO_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsinfo")

/*
 * Resource dispatch and information tables. Any new resource types (either
 * Large or Small) must be reflected in each of these tables, so they are here
 * in one place.
 *
 * The tables for Large descriptors are indexed by bits 6:0 of the AML
 * descriptor type byte. The tables for Small descriptors are indexed by
```



```
 * bits 6:3 of the descriptor byte. The tables for internal resource
 * descriptors are indexed by the ACPI_RESOURCE_TYPE field.
 */


/* Dispatch table for resource-to-AML (Set Resource) conversion functions */

ACPI_RSCONVERT_INFO         *AcpiGbl_SetResourceDispatch[] =
{
    AcpiRsSetIrq,                   /* 0x00, ACPI_RESOURCE_TYPE_IRQ */
    AcpiRsConvertDma,               /* 0x01, ACPI_RESOURCE_TYPE_DMA */
    AcpiRsSetStartDpf,              /* 0x02, ACPI_RESOURCE_TYPE_START_DEPENDENT
    AcpiRsConvertEndDpf,            /* 0x03, ACPI_RESOURCE_TYPE_END_DEPENDENT */
    AcpiRsConvertIo,                /* 0x04, ACPI_RESOURCE_TYPE_IO */
    AcpiRsConvertFixedIo,           /* 0x05, ACPI_RESOURCE_TYPE_FIXED_IO */
    AcpiRsSetVendor,                /* 0x06, ACPI_RESOURCE_TYPE_VENDOR */
    AcpiRsConvertEndTag,            /* 0x07, ACPI_RESOURCE_TYPE_END_TAG */
    AcpiRsConvertMemory24,          /* 0x08, ACPI_RESOURCE_TYPE_MEMORY24 */
    AcpiRsConvertMemory32,          /* 0x09, ACPI_RESOURCE_TYPE_MEMORY32 */
    AcpiRsConvertFixedMemory32,     /* 0x0A, ACPI_RESOURCE_TYPE_FIXED_MEMORY32 *
    AcpiRsConvertAddress16,         /* 0x0B, ACPI_RESOURCE_TYPE_ADDRESS16 */
    AcpiRsConvertAddress32,         /* 0x0C, ACPI_RESOURCE_TYPE_ADDRESS32 */
    AcpiRsConvertAddress64,         /* 0x0D, ACPI_RESOURCE_TYPE_ADDRESS64 */
    AcpiRsConvertExtAddress64,      /* 0x0E, ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS
    AcpiRsConvertExtIrq,            /* 0x0F, ACPI_RESOURCE_TYPE_EXTENDED_IRQ */
    AcpiRsConvertGenericReg,        /* 0x10, ACPI_RESOURCE_TYPE_GENERIC_REGISTER
    AcpiRsConvertGpio,              /* 0x11, ACPI_RESOURCE_TYPE_GPIO */
    AcpiRsConvertFixedDma,          /* 0x12, ACPI_RESOURCE_TYPE_FIXED_DMA */
    NULL,                           /* 0x13, ACPI_RESOURCE_TYPE_SERIAL_BUS - Use
    AcpiRsConvertGenericReg         /* 0x10, ACPI_RESOURCE_TYPE_GENERIC_REGISTER
};

/* Dispatch tables for AML-to-resource (Get Resource) conversion functions */

ACPI_RSCONVERT_INFO         *AcpiGbl_GetResourceDispatch[] =
{
    /* Small descriptors */

    NULL,                           /* 0x00, Reserved */
    NULL,                           /* 0x01, Reserved */
    NULL,                           /* 0x02, Reserved */
    NULL,                           /* 0x03, Reserved */
    AcpiRsGetIrq,                   /* 0x04, ACPI_RESOURCE_NAME_IRQ */
    AcpiRsConvertDma,               /* 0x05, ACPI_RESOURCE_NAME_DMA */
    AcpiRsGetStartDpf,              /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT
    AcpiRsConvertEndDpf,            /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
    AcpiRsConvertIo,                /* 0x08, ACPI_RESOURCE_NAME_IO */
    AcpiRsConvertFixedIo,           /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO */
    AcpiRsConvertFixedDma,          /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
    NULL,                           /* 0x0A, Reserved */
    NULL,                           /* 0x0B, Reserved */
    NULL,                           /* 0x0C, Reserved */
    NULL,                           /* 0x0D, Reserved */
    AcpiRsGetVendorSmall,           /* 0x0E, ACPI_RESOURCE_NAME_VENDOR_SMALL */
    AcpiRsConvertEndTag,            /* 0x0F, ACPI_RESOURCE_NAME_END_TAG */

    /* Large descriptors */

    NULL,                           /* 0x00, Reserved */
    AcpiRsConvertMemory24,          /* 0x01, ACPI_RESOURCE_NAME_MEMORY24 */
    AcpiRsConvertGenericReg,        /* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER
    NULL,                           /* 0x03, Reserved */
    AcpiRsGetVendorLarge,           /* 0x04, ACPI_RESOURCE_NAME_VENDOR_LARGE */
    AcpiRsConvertMemory32,          /* 0x05, ACPI_RESOURCE_NAME_MEMORY32 */
    AcpiRsConvertFixedMemory32,     /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY32 *
    AcpiRsConvertAddress32,         /* 0x07, ACPI_RESOURCE_NAME_ADDRESS32 */
```

```
    AcpiRsConvertAddress16,         /* 0x08, ACPI_RESOURCE_NAME_ADDRESS16 */
    AcpiRsConvertExtIrq,            /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_IRQ */
    AcpiRsConvertAddress64,         /* 0x0A, ACPI_RESOURCE_NAME_ADDRESS64 */
    AcpiRsConvertExtAddress64,      /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
    AcpiRsConvertGpio,              /* 0x0C, ACPI_RESOURCE_NAME_GPIO */
    NULL,                           /* 0x0D, Reserved */
    NULL,                           /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS - Use
    AcpiRsConvertExtAddress64       /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
};

/* Subtype table for SerialBus -- I2C, SPI, and UART */

ACPI_RSCONVERT_INFO         *AcpiGbl_ConvertResourceSerialBusDispatch[] =
{
    NULL,
    AcpiRsConvertI2cSerialBus,
    AcpiRsConvertSpiSerialBus,
    AcpiRsConvertUartSerialBus,
};


#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

/* Dispatch table for resource dump functions */

ACPI_RSDUMP_INFO            *AcpiGbl_DumpResourceDispatch[] =
{
    AcpiRsDumpIrq,                  /* ACPI_RESOURCE_TYPE_IRQ */
    AcpiRsDumpDma,                  /* ACPI_RESOURCE_TYPE_DMA */
    AcpiRsDumpStartDpf,             /* ACPI_RESOURCE_TYPE_START_DEPENDENT */
    AcpiRsDumpEndDpf,               /* ACPI_RESOURCE_TYPE_END_DEPENDENT */
    AcpiRsDumpIo,                   /* ACPI_RESOURCE_TYPE_IO */
    AcpiRsDumpFixedIo,              /* ACPI_RESOURCE_TYPE_FIXED_IO */
    AcpiRsDumpVendor,               /* ACPI_RESOURCE_TYPE_VENDOR */
    AcpiRsDumpEndTag,               /* ACPI_RESOURCE_TYPE_END_TAG */
    AcpiRsDumpMemory24,             /* ACPI_RESOURCE_TYPE_MEMORY24 */
    AcpiRsDumpMemory32,             /* ACPI_RESOURCE_TYPE_MEMORY32 */
    AcpiRsDumpFixedMemory32,        /* ACPI_RESOURCE_TYPE_FIXED_MEMORY32 */
    AcpiRsDumpAddress16,            /* ACPI_RESOURCE_TYPE_ADDRESS16 */
    AcpiRsDumpAddress32,            /* ACPI_RESOURCE_TYPE_ADDRESS32 */
    AcpiRsDumpAddress64,            /* ACPI_RESOURCE_TYPE_ADDRESS64 */
    AcpiRsDumpExtAddress64,         /* ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64 */
    AcpiRsDumpExtIrq,               /* ACPI_RESOURCE_TYPE_EXTENDED_IRQ */
    AcpiRsDumpGenericReg,           /* ACPI_RESOURCE_TYPE_GENERIC_REGISTER */
    AcpiRsDumpGpio,                 /* ACPI_RESOURCE_TYPE_GPIO */
    AcpiRsDumpFixedDma,             /* ACPI_RESOURCE_TYPE_FIXED_DMA */
    NULL,                           /* ACPI_RESOURCE_TYPE_SERIAL_BUS */
};

ACPI_RSDUMP_INFO            *AcpiGbl_DumpSerialBusDispatch[] =
{
    NULL,
    AcpiRsDumpI2cSerialBus,         /* AML_RESOURCE_I2C_BUS_TYPE */
    AcpiRsDumpSpiSerialBus,         /* AML_RESOURCE_SPI_BUS_TYPE */
    AcpiRsDumpUartSerialBus,        /* AML_RESOURCE_UART_BUS_TYPE */
};
#endif


/*
 * Base sizes for external AML resource descriptors, indexed by internal type.
 * Includes size of the descriptor header (1 byte for small descriptors,
 * 3 bytes for large descriptors)
 */
const UINT8                 AcpiGbl_AmlResourceSizes[] =
{
```



```
    sizeof (AML_RESOURCE_IRQ),              /* ACPI_RESOURCE_TYPE_IRQ (optional
    sizeof (AML_RESOURCE_DMA),              /* ACPI_RESOURCE_TYPE_DMA */
    sizeof (AML_RESOURCE_START_DEPENDENT),  /* ACPI_RESOURCE_TYPE_START_DEPENDEN
    sizeof (AML_RESOURCE_END_DEPENDENT),    /* ACPI_RESOURCE_TYPE_END_DEPENDENT
    sizeof (AML_RESOURCE_IO),               /* ACPI_RESOURCE_TYPE_IO */
    sizeof (AML_RESOURCE_FIXED_IO),         /* ACPI_RESOURCE_TYPE_FIXED_IO */
    sizeof (AML_RESOURCE_VENDOR_SMALL),     /* ACPI_RESOURCE_TYPE_VENDOR */
    sizeof (AML_RESOURCE_END_TAG),          /* ACPI_RESOURCE_TYPE_END_TAG */
    sizeof (AML_RESOURCE_MEMORY24),         /* ACPI_RESOURCE_TYPE_MEMORY24 */
    sizeof (AML_RESOURCE_MEMORY32),         /* ACPI_RESOURCE_TYPE_MEMORY32 */
    sizeof (AML_RESOURCE_FIXED_MEMORY32),   /* ACPI_RESOURCE_TYPE_FIXED_MEMORY32
    sizeof (AML_RESOURCE_ADDRESS16),        /* ACPI_RESOURCE_TYPE_ADDRESS16 */
    sizeof (AML_RESOURCE_ADDRESS32),        /* ACPI_RESOURCE_TYPE_ADDRESS32 */
    sizeof (AML_RESOURCE_ADDRESS64),        /* ACPI_RESOURCE_TYPE_ADDRESS64 */
    sizeof (AML_RESOURCE_EXTENDED_ADDRESS64),/*ACPI_RESOURCE_TYPE_EXTENDED_ADDRE
    sizeof (AML_RESOURCE_EXTENDED_IRQ),     /* ACPI_RESOURCE_TYPE_EXTENDED_IRQ *
    sizeof (AML_RESOURCE_GENERIC_REGISTER), /* ACPI_RESOURCE_TYPE_GENERIC_REGIST
    sizeof (AML_RESOURCE_GPIO),             /* ACPI_RESOURCE_TYPE_GPIO */
    sizeof (AML_RESOURCE_FIXED_DMA),        /* ACPI_RESOURCE_TYPE_FIXED_DMA */
    sizeof (AML_RESOURCE_COMMON_SERIALBUS), /* ACPI_RESOURCE_TYPE_SERIAL_BUS */
    sizeof (AML_RESOURCE_GENERIC_REGISTER)  /* ACPI_RESOURCE_TYPE_GENERIC_REGIST
};


const UINT8                 AcpiGbl_ResourceStructSizes[] =
{
    /* Small descriptors */

    0,
    0,
    0,
    0,
    ACPI_RS_SIZE (ACPI_RESOURCE_IRQ),
    ACPI_RS_SIZE (ACPI_RESOURCE_DMA),
    ACPI_RS_SIZE (ACPI_RESOURCE_START_DEPENDENT),
    ACPI_RS_SIZE_MIN,
    ACPI_RS_SIZE (ACPI_RESOURCE_IO),
    ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_IO),
    ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_DMA),
    0,
    0,
    0,
    0,
    ACPI_RS_SIZE (ACPI_RESOURCE_VENDOR),
    ACPI_RS_SIZE_MIN,

    /* Large descriptors */

    0,
    ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY24),
    ACPI_RS_SIZE (ACPI_RESOURCE_GENERIC_REGISTER),
    0,
    ACPI_RS_SIZE (ACPI_RESOURCE_VENDOR),
    ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY32),
    ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_MEMORY32),
    ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS32),
    ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS16),
    ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_IRQ),
    ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS64),
    ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_ADDRESS64),
    ACPI_RS_SIZE (ACPI_RESOURCE_GPIO),
    ACPI_RS_SIZE (ACPI_RESOURCE_COMMON_SERIALBUS)
    ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_ADDRESS64)
};

const UINT8                 AcpiGbl_AmlResourceSerialBusSizes[] =
```

```
252 {
253     0,
254     sizeof (AML_RESOURCE_I2C_SERIALBUS),
255     sizeof (AML_RESOURCE_SPI_SERIALBUS),
256     sizeof (AML_RESOURCE_UART_SERIALBUS),
257 };

259 const UINT8                AcpiGbl_ResourceStructSerialBusSizes[] =
260 {
261     0,
262     ACPI_RS_SIZE (ACPI_RESOURCE_I2C_SERIALBUS),
263     ACPI_RS_SIZE (ACPI_RESOURCE_SPI_SERIALBUS),
264     ACPI_RS_SIZE (ACPI_RESOURCE_UART_SERIALBUS),
265 };
```

****
10605 Thu Dec 26 13:49:30 2013
new/usr/src/common/acpica/components/resources/rsio.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: rsio - IO and DMA resource descriptors
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __RSIO_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acresrc.h"

  50 #define _COMPONENT          ACPI_RESOURCES
  51          ACPI_MODULE_NAME    ("rsio")


  54 /******************************************************************************
  55  *
  56  * AcpiRsConvertIo
  57  *
  58  *****************************************************************************/
```



```
  60 ACPI_RSCONVERT_INFO     AcpiRsConvertIo[5] =
  61 {
  62     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_IO,
  63                         ACPI_RS_SIZE (ACPI_RESOURCE_IO),
  64                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertIo)},

  66     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_IO,
  67                         sizeof (AML_RESOURCE_IO),
  68                         0},

  70     /* Decode flag */

  72     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Io.IoDecode),
  73                         AML_OFFSET (Io.Flags),
  74                         0},
  75     /*
  76      * These fields are contiguous in both the source and destination:
  77      * Address Alignment
  78      * Length
  79      * Minimum Base Address
  80      * Maximum Base Address
  81      */
  82     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.Io.Alignment),
  83                         AML_OFFSET (Io.Alignment),
  84                         2},

  86     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.Io.Minimum),
  87                         AML_OFFSET (Io.Minimum),
  88                         2}
  89 };
```

_____unchanged_portion_omitted_

```
**********************************************************
   11371 Thu Dec 26 13:49:30 2013
new/usr/src/common/acpica/components/resources/rsirq.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsirq - IRQ resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSIRQ_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsirq")


/******************************************************************************
 *
 * AcpiRsGetIrq
 *
 *****************************************************************************/
```



```
ACPI_RSCONVERT_INFO     AcpiRsGetIrq[9] =
ACPI_RSCONVERT_INFO     AcpiRsGetIrq[8] =
{
    {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_IRQ,
                        ACPI_RS_SIZE (ACPI_RESOURCE_IRQ),
                        ACPI_RSC_TABLE_SIZE (AcpiRsGetIrq)},

    /* Get the IRQ mask (bytes 1:2) */

    {ACPI_RSC_BITMASK16,ACPI_RS_OFFSET (Data.Irq.Interrupts[0]),
                        AML_OFFSET (Irq.IrqMask),
                        ACPI_RS_OFFSET (Data.Irq.InterruptCount)},

    /* Set default flags (others are zero) */

    {ACPI_RSC_SET8,     ACPI_RS_OFFSET (Data.Irq.Triggering),
                        ACPI_EDGE_SENSITIVE,
                        1},

    /* Get the descriptor length (2 or 3 for IRQ descriptor) */

    {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.Irq.DescriptorLength),
                        AML_OFFSET (Irq.DescriptorType),
                        0},

    /* All done if no flag byte present in descriptor */

    {ACPI_RSC_EXIT_NE,  ACPI_RSC_COMPARE_AML_LENGTH, 0, 3},

    /* Get flags: Triggering[0], Polarity[3], Sharing[4], Wake[5] */
    /* Get flags: Triggering[0], Polarity[3], Sharing[4] */

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Triggering),
                        AML_OFFSET (Irq.Flags),
                        0},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Polarity),
                        AML_OFFSET (Irq.Flags),
                        3},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Sharable),
                        AML_OFFSET (Irq.Flags),
                        4},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.WakeCapable),
                        AML_OFFSET (Irq.Flags),
                        5}
                        4}
};


/******************************************************************************
 *
 * AcpiRsSetIrq
 *
 *****************************************************************************/

ACPI_RSCONVERT_INFO     AcpiRsSetIrq[14] =
ACPI_RSCONVERT_INFO     AcpiRsSetIrq[13] =
{
    /* Start with a default descriptor of length 3 */

    {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_IRQ,
                        sizeof (AML_RESOURCE_IRQ),
                        ACPI_RSC_TABLE_SIZE (AcpiRsSetIrq)},
```

```
122     /* Convert interrupt list to 16-bit IRQ bitmask */
123
124     {ACPI_RSC_BITMASK16,ACPI_RS_OFFSET (Data.Irq.Interrupts[0]),
125                         AML_OFFSET (Irq.IrqMask),
126                         ACPI_RS_OFFSET (Data.Irq.InterruptCount)},
127
128     /* Set flags: Triggering[0], Polarity[3], Sharing[4], Wake[5] */
124     /* Set the flags byte */
129
130     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Triggering),
131                         AML_OFFSET (Irq.Flags),
132                         0},
133
134     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Polarity),
135                         AML_OFFSET (Irq.Flags),
136                         3},
137
138     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Sharable),
139                         AML_OFFSET (Irq.Flags),
140                         4},
141
142     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.WakeCapable),
143                         AML_OFFSET (Irq.Flags),
144                         5},
145
146     /*
147      * All done if the output descriptor length is required to be 3
148      * (i.e., optimization to 2 bytes cannot be attempted)
149      */
150     {ACPI_RSC_EXIT_EQ,  ACPI_RSC_COMPARE_VALUE,
151                         ACPI_RS_OFFSET(Data.Irq.DescriptorLength),
152                         3},
153
154     /* Set length to 2 bytes (no flags byte) */
155
156     {ACPI_RSC_LENGTH,   0, 0, sizeof (AML_RESOURCE_IRQ_NOFLAGS)},
157
158     /*
159      * All done if the output descriptor length is required to be 2.
160      *
161      * TBD: Perhaps we should check for error if input flags are not
162      * compatible with a 2-byte descriptor.
163      */
164     {ACPI_RSC_EXIT_EQ,  ACPI_RSC_COMPARE_VALUE,
165                         ACPI_RS_OFFSET(Data.Irq.DescriptorLength),
166                         2},
167
168     /* Reset length to 3 bytes (descriptor with flags byte) */
169
170     {ACPI_RSC_LENGTH,   0, 0, sizeof (AML_RESOURCE_IRQ)},
171
172     /*
173      * Check if the flags byte is necessary. Not needed if the flags are:
174      * ACPI_EDGE_SENSITIVE, ACPI_ACTIVE_HIGH, ACPI_EXCLUSIVE
175      */
176     {ACPI_RSC_EXIT_NE,  ACPI_RSC_COMPARE_VALUE,
177                         ACPI_RS_OFFSET (Data.Irq.Triggering),
178                         ACPI_EDGE_SENSITIVE},
179
180     {ACPI_RSC_EXIT_NE,  ACPI_RSC_COMPARE_VALUE,
181                         ACPI_RS_OFFSET (Data.Irq.Polarity),
182                         ACPI_ACTIVE_HIGH},
183
184     {ACPI_RSC_EXIT_NE,  ACPI_RSC_COMPARE_VALUE,
185                         ACPI_RS_OFFSET (Data.Irq.Sharable),
186                         ACPI_EXCLUSIVE},
```



```
187
188     /* We can optimize to a 2-byte IrqNoFlags() descriptor */
189
190     {ACPI_RSC_LENGTH,   0, 0, sizeof (AML_RESOURCE_IRQ_NOFLAGS)}
191 };
192
193
194 /*******************************************************************************
195  *
196  * AcpiRsConvertExtIrq
197  *
198  ******************************************************************************/
199
200 ACPI_RSCONVERT_INFO     AcpiRsConvertExtIrq[10] =
192 ACPI_RSCONVERT_INFO     AcpiRsConvertExtIrq[9] =
201 {
202     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_EXTENDED_IRQ,
203                         ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_IRQ),
204                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertExtIrq)},
205
206     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_EXTENDED_IRQ,
207                         sizeof (AML_RESOURCE_EXTENDED_IRQ),
208                         0},
209
210     /*
211      * Flags: Producer/Consumer[0], Triggering[1], Polarity[2],
212      *        Sharing[3], Wake[4]
213      */
202     /* Flag bits */
214     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.ProducerConsumer),
215                         AML_OFFSET (ExtendedIrq.Flags),
216                         0},
217
218     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Triggering),
219                         AML_OFFSET (ExtendedIrq.Flags),
220                         1},
221
222     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Polarity),
223                         AML_OFFSET (ExtendedIrq.Flags),
224                         2},
225
226     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Sharable),
227                         AML_OFFSET (ExtendedIrq.Flags),
228                         3},
229
230     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.WakeCapable),
231                         AML_OFFSET (ExtendedIrq.Flags),
232                         4},
233
234     /* IRQ Table length (Byte4) */
235
236     {ACPI_RSC_COUNT,    ACPI_RS_OFFSET (Data.ExtendedIrq.InterruptCount),
237                         AML_OFFSET (ExtendedIrq.InterruptCount),
238                         sizeof (UINT32)},
239
240     /* Copy every IRQ in the table, each is 32 bits */
241
242     {ACPI_RSC_MOVE32,   ACPI_RS_OFFSET (Data.ExtendedIrq.Interrupts[0]),
243                         AML_OFFSET (ExtendedIrq.Interrupts[0]),
244                         0},
245
246     /* Optional ResourceSource (Index and String) */
247
248     {ACPI_RSC_SOURCEX,  ACPI_RS_OFFSET (Data.ExtendedIrq.ResourceSource),
249                         ACPI_RS_OFFSET (Data.ExtendedIrq.Interrupts[0]),
```

```
250                         sizeof (AML_RESOURCE_EXTENDED_IRQ)}
251 };
```

*unchanged_portion_omitted_*

```
292 /*******************************************************************************
293  *
294  * AcpiRsConvertFixedDma
295  *
296  ******************************************************************************/
298 ACPI_RSCONVERT_INFO     AcpiRsConvertFixedDma[4] =
299 {
300     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_FIXED_DMA,
301                         ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_DMA),
302                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertFixedDma)},
304     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_FIXED_DMA,
305                         sizeof (AML_RESOURCE_FIXED_DMA),
306                         0},
308     /*
309      * These fields are contiguous in both the source and destination:
310      * RequestLines
311      * Channels
312      */
313     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.FixedDma.RequestLines),
314                         AML_OFFSET (FixedDma.RequestLines),
315                         2},
317     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.FixedDma.Width),
318                         AML_OFFSET (FixedDma.Width),
319                         1},
320 };
```

```
**********************************************************
    9588 Thu Dec 26 13:49:31 2013
new/usr/src/common/acpica/components/resources/rslist.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rslist - Linked list utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSLIST_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rslist")


/******************************************************************************
 *
 * FUNCTION:    AcpiRsConvertAmlToResources
 *
 * PARAMETERS:  ACPI_WALK_AML_CALLBACK
 *              ResourcePtr             - Pointer to the buffer that will
 *                                        contain the output structures
```



```
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert an AML resource to an internal representation of the
 *              resource that is aligned and easier to access.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiRsConvertAmlToResources (
    UINT8                   *Aml,
    UINT32                  Length,
    UINT32                  Offset,
    UINT8                   ResourceIndex,
    void                    **Context)
{
    ACPI_RESOURCE           **ResourcePtr = ACPI_CAST_INDIRECT_PTR (
                                ACPI_RESOURCE, Context);
    ACPI_RESOURCE           *Resource;
    AML_RESOURCE            *AmlResource;
    ACPI_RSCONVERT_INFO     *ConversionTable;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (RsConvertAmlToResources);


    /*
     * Check that the input buffer and all subsequent pointers into it
     * are aligned on a native word boundary. Most important on IA64
     */
    Resource = *ResourcePtr;
    if (ACPI_IS_MISALIGNED (Resource))
    {
        ACPI_WARNING ((AE_INFO,
            "Misaligned resource pointer %p", Resource));
    }

    /* Get the appropriate conversion info table */

    AmlResource = ACPI_CAST_PTR (AML_RESOURCE, Aml);
    if (AcpiUtGetResourceType (Aml) == ACPI_RESOURCE_NAME_SERIAL_BUS)
    {
        if (AmlResource->CommonSerialBus.Type > AML_RESOURCE_MAX_SERIALBUSTYPE)
        {
            ConversionTable = NULL;
        }
        else
        {
            /* This is an I2C, SPI, or UART SerialBus descriptor */

            ConversionTable =
                AcpiGbl_ConvertResourceSerialBusDispatch[
                    AmlResource->CommonSerialBus.Type];
        }
    }
    else
    {
        ConversionTable =
            AcpiGbl_GetResourceDispatch[ResourceIndex];
    }

    if (!ConversionTable)
    {
        ACPI_ERROR ((AE_INFO,
            "Invalid/unsupported resource descriptor: Type 0x%2.2X",
```

```
127             ResourceIndex));
128         return_ACPI_STATUS (AE_AML_INVALID_RESOURCE_TYPE);
129     }

131      /* Convert the AML byte stream resource to a local resource struct */

133     Status = AcpiRsConvertAmlToResource (
134         Resource, AmlResource, ConversionTable);
135     if (ACPI_FAILURE (Status))
136     {
137         ACPI_EXCEPTION ((AE_INFO, Status,
138             "Could not convert AML resource (Type 0x%X)", *Aml));
139         return_ACPI_STATUS (Status);
140     }

142     ACPI_DEBUG_PRINT ((ACPI_DB_RESOURCES,
143         "Type %.2X, AmlLength %.2X InternalLength %.2X\n",
144         AcpiUtGetResourceType (Aml), Length,
145         Resource->Length));

147     /* Point to the next structure in the output buffer */

149     *ResourcePtr = ACPI_NEXT_RESOURCE (Resource);
150     return_ACPI_STATUS (AE_OK);
151 }


154 /******************************************************************************
155  *
156  * FUNCTION:    AcpiRsConvertResourcesToAml
157  *
158  * PARAMETERS:  Resource            - Pointer to the resource linked list
159  *              AmlSizeNeeded       - Calculated size of the byte stream
160  *                                    needed from calling AcpiRsGetAmlLength()
161  *                                    The size of the OutputBuffer is
162  *                                    guaranteed to be >= AmlSizeNeeded
163  *              OutputBuffer        - Pointer to the buffer that will
164  *                                    contain the byte stream
165  *
166  * RETURN:      Status
167  *
168  * DESCRIPTION: Takes the resource linked list and parses it, creating a
169  *              byte stream of resources in the caller's output buffer
170  *
171  *****************************************************************************/

173 ACPI_STATUS
174 AcpiRsConvertResourcesToAml (
175     ACPI_RESOURCE           *Resource,
176     ACPI_SIZE               AmlSizeNeeded,
177     UINT8                   *OutputBuffer)
178 {
179     UINT8                   *Aml = OutputBuffer;
180     UINT8                   *EndAml = OutputBuffer + AmlSizeNeeded;
181     ACPI_RSCONVERT_INFO     *ConversionTable;
182     ACPI_STATUS             Status;


185     ACPI_FUNCTION_TRACE (RsConvertResourcesToAml);


188     /* Walk the resource descriptor list, convert each descriptor */

190     while (Aml < EndAml)
191     {
192         /* Validate the (internal) Resource Type */
```



```
194         if (Resource->Type > ACPI_RESOURCE_TYPE_MAX)
195         {
196             ACPI_ERROR ((AE_INFO,
197                 "Invalid descriptor type (0x%X) in resource list",
198                 Resource->Type));
199             return_ACPI_STATUS (AE_BAD_DATA);
200         }

202         /* Sanity check the length. It must not be zero, or we loop forever */

204         if (!Resource->Length)
205         {
206             ACPI_ERROR ((AE_INFO,
207                 "Invalid zero length descriptor in resource list\n"));
208             return_ACPI_STATUS (AE_AML_BAD_RESOURCE_LENGTH);
209         }

211         /* Perform the conversion */

213         if (Resource->Type == ACPI_RESOURCE_TYPE_SERIAL_BUS)
214         {
215             if (Resource->Data.CommonSerialBus.Type > AML_RESOURCE_MAX_SERIALBUS
216             {
217                 ConversionTable = NULL;
218             }
219             else
220             {
221                 /* This is an I2C, SPI, or UART SerialBus descriptor */

223                 ConversionTable = AcpiGbl_ConvertResourceSerialBusDispatch[
224                     Resource->Data.CommonSerialBus.Type];
225             }
226         }
227         else
228         {
229             ConversionTable = AcpiGbl_SetResourceDispatch[Resource->Type];
230         }

232         if (!ConversionTable)
233         {
234             ACPI_ERROR ((AE_INFO,
235                 "Invalid/unsupported resource descriptor: Type 0x%2.2X",
236                 Resource->Type));
237             return_ACPI_STATUS (AE_AML_INVALID_RESOURCE_TYPE);
238         }

240         Status = AcpiRsConvertResourceToAml (Resource,
241                 ACPI_CAST_PTR (AML_RESOURCE, Aml),
242                 ConversionTable);
243         if (ACPI_FAILURE (Status))
244         {
245             ACPI_EXCEPTION ((AE_INFO, Status,
246                 "Could not convert resource (type 0x%X) to AML",
247                 Resource->Type));
248             return_ACPI_STATUS (Status);
249         }

251         /* Perform final sanity check on the new AML resource descriptor */

253         Status = AcpiUtValidateResource (NULL,
254                     ACPI_CAST_PTR (AML_RESOURCE, Aml), NULL);
255         if (ACPI_FAILURE (Status))
256         {
257             return_ACPI_STATUS (Status);
258         }
```

```
260         /* Check for end-of-list, normal exit */

262         if (Resource->Type == ACPI_RESOURCE_TYPE_END_TAG)
263         {
264             /* An End Tag indicates the end of the input Resource Template */

266             return_ACPI_STATUS (AE_OK);
267         }

269         /*
270          * Extract the total length of the new descriptor and set the
271          * Aml to point to the next (output) resource descriptor
272          */
273         Aml += AcpiUtGetDescriptorLength (Aml);

275         /* Point to the next input resource descriptor */

277         Resource = ACPI_NEXT_RESOURCE (Resource);
278     }

280     /* Completed buffer, but did not find an EndTag resource descriptor */

282     return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
283 }
```

****
8546 Thu Dec 26 13:49:31 2013
new/usr/src/common/acpica/components/resources/rsmemory.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: rsmem24 - Memory resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSMEMORY_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsmemory")


/******************************************************************************
 *
 * AcpiRsConvertMemory24
 *
 *****************************************************************************/
```



```
ACPI_RSCONVERT_INFO     AcpiRsConvertMemory24[4] =
{
    {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_MEMORY24,
                        ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY24),
                        ACPI_RSC_TABLE_SIZE (AcpiRsConvertMemory24)},

    {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_MEMORY24,
                        sizeof (AML_RESOURCE_MEMORY24),
                        0},

    /* Read/Write bit */

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Memory24.WriteProtect),
                        AML_OFFSET (Memory24.Flags),
                        0},
    /*
     * These fields are contiguous in both the source and destination:
     * Minimum Base Address
     * Maximum Base Address
     * Address Base Alignment
     * Range Length
     */
    {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.Memory24.Minimum),
                        AML_OFFSET (Memory24.Minimum),
                        4}
};
```

_____unchanged_portion_omitted_

```
**********************************************************
   25173 Thu Dec 26 13:49:31 2013
new/usr/src/common/acpica/components/resources/rsmisc.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsmisc - Miscellaneous resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSMISC_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsmisc")


#define INIT_RESOURCE_TYPE(i)       i->ResourceOffset
#define INIT_RESOURCE_LENGTH(i)     i->AmlOffset
#define INIT_TABLE_LENGTH(i)        i->Value

#define COMPARE_OPCODE(i)           i->ResourceOffset
#define COMPARE_TARGET(i)           i->AmlOffset
#define COMPARE_VALUE(i)            i->Value
```



```
/*******************************************************************************
 *
 * FUNCTION:    AcpiRsConvertAmlToResource
 *
 * PARAMETERS:  Resource            - Pointer to the resource descriptor
 *              Aml                 - Where the AML descriptor is returned
 *              Info                - Pointer to appropriate conversion table
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert an external AML resource descriptor to the corresponding
 *              internal resource descriptor
 *
 ******************************************************************************/

ACPI_STATUS
AcpiRsConvertAmlToResource (
    ACPI_RESOURCE           *Resource,
    AML_RESOURCE            *Aml,
    ACPI_RSCONVERT_INFO     *Info)
{
    ACPI_RS_LENGTH          AmlResourceLength;
    void                    *Source;
    void                    *Destination;
    char                    *Target;
    UINT8                   Count;
    UINT8                   FlagsMode = FALSE;
    UINT16                  ItemCount = 0;
    UINT16                  Temp16 = 0;


    ACPI_FUNCTION_TRACE (RsConvertAmlToResource);


    if (!Info)
    {
        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (((ACPI_SIZE) Resource) & 0x3)
    {
        /* Each internal resource struct is expected to be 32-bit aligned */

        ACPI_WARNING ((AE_INFO,
            "Misaligned resource pointer (get): %p Type 0x%2.2X Length %u",
            Resource, Resource->Type, Resource->Length));
    }

    /* Extract the resource Length field (does not include header length) */

    AmlResourceLength = AcpiUtGetResourceLength (Aml);

    /*
     * First table entry must be ACPI_RSC_INITxxx and must contain the
     * table length (# of table entries)
     */
    Count = INIT_TABLE_LENGTH (Info);
    while (Count)
    {
        /*
         * Source is the external AML byte stream buffer,
         * destination is the internal resource descriptor
         */
        Source      = ACPI_ADD_PTR (void, Aml, Info->AmlOffset);
```

```
127         Destination = ACPI_ADD_PTR (void, Resource, Info->ResourceOffset);

129         switch (Info->Opcode)
130         {
131         case ACPI_RSC_INITGET:
132             /*
133              * Get the resource type and the initial (minimum) length
134              */
135             ACPI_MEMSET (Resource, 0, INIT_RESOURCE_LENGTH (Info));
136             Resource->Type = INIT_RESOURCE_TYPE (Info);
137             Resource->Length = INIT_RESOURCE_LENGTH (Info);
138             break;

140         case ACPI_RSC_INITSET:
141             break;

143         case ACPI_RSC_FLAGINIT:

145             FlagsMode = TRUE;
146             break;

148         case ACPI_RSC_1BITFLAG:
149             /*
150              * Mask and shift the flag bit
151              */
152             ACPI_SET8 (Destination,
153                 ((ACPI_GET8 (Source) >> Info->Value) & 0x01));
154             break;

156         case ACPI_RSC_2BITFLAG:
157             /*
158              * Mask and shift the flag bits
159              */
160             ACPI_SET8 (Destination,
161                 ((ACPI_GET8 (Source) >> Info->Value) & 0x03));
162             break;

164         case ACPI_RSC_3BITFLAG:
165             /*
166              * Mask and shift the flag bits
167              */
168             ACPI_SET8 (Destination,
169                 ((ACPI_GET8 (Source) >> Info->Value) & 0x07));
170             break;

172         case ACPI_RSC_COUNT:

174             ItemCount = ACPI_GET8 (Source);
175             ACPI_SET8 (Destination, ItemCount);

177             Resource->Length = Resource->Length +
178                 (Info->Value * (ItemCount - 1));
179             break;

181         case ACPI_RSC_COUNT16:

183             ItemCount = AmlResourceLength;
184             ACPI_SET16 (Destination, ItemCount);

186             Resource->Length = Resource->Length +
187                 (Info->Value * (ItemCount - 1));
188             break;

190         case ACPI_RSC_COUNT_GPIO_PIN:

192             Target = ACPI_ADD_PTR (void, Aml, Info->Value);
```



```
193             ItemCount = ACPI_GET16 (Target) - ACPI_GET16 (Source);

195             Resource->Length = Resource->Length + ItemCount;
196             ItemCount = ItemCount / 2;
197             ACPI_SET16 (Destination, ItemCount);
198             break;

200         case ACPI_RSC_COUNT_GPIO_VEN:

202             ItemCount = ACPI_GET8 (Source);
203             ACPI_SET8 (Destination, ItemCount);

205             Resource->Length = Resource->Length +
206                 (Info->Value * ItemCount);
207             break;

209         case ACPI_RSC_COUNT_GPIO_RES:
210             /*
211              * Vendor data is optional (length/offset may both be zero)
212              * Examine vendor data length field first
213              */
214             Target = ACPI_ADD_PTR (void, Aml, (Info->Value + 2));
215             if (ACPI_GET16 (Target))
216             {
217                 /* Use vendor offset to get resource source length */

219                 Target = ACPI_ADD_PTR (void, Aml, Info->Value);
220                 ItemCount = ACPI_GET16 (Target) - ACPI_GET16 (Source);
221             }
222             else
223             {
224                 /* No vendor data to worry about */

226                 ItemCount = Aml->LargeHeader.ResourceLength +
227                     sizeof (AML_RESOURCE_LARGE_HEADER) -
228                     ACPI_GET16 (Source);
229             }

231             Resource->Length = Resource->Length + ItemCount;
232             ACPI_SET16 (Destination, ItemCount);
233             break;

235         case ACPI_RSC_COUNT_SERIAL_VEN:

237             ItemCount = ACPI_GET16 (Source) - Info->Value;

239             Resource->Length = Resource->Length + ItemCount;
240             ACPI_SET16 (Destination, ItemCount);
241             break;

243         case ACPI_RSC_COUNT_SERIAL_RES:

245             ItemCount = (AmlResourceLength +
246                 sizeof (AML_RESOURCE_LARGE_HEADER)) -
247                 ACPI_GET16 (Source) - Info->Value;

249             Resource->Length = Resource->Length + ItemCount;
250             ACPI_SET16 (Destination, ItemCount);
251             break;

253         case ACPI_RSC_LENGTH:

255             Resource->Length = Resource->Length + Info->Value;
256             break;

258         case ACPI_RSC_MOVE8:
```

```
259         case ACPI_RSC_MOVE16:
260         case ACPI_RSC_MOVE32:
261         case ACPI_RSC_MOVE64:
262             /*
263              * Raw data move. Use the Info value field unless ItemCount has
264              * been previously initialized via a COUNT opcode
265              */
266             if (Info->Value)
267             {
268                 ItemCount = Info->Value;
269             }
270             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
271             break;

273         case ACPI_RSC_MOVE_GPIO_PIN:

275             /* Generate and set the PIN data pointer */

277             Target = (char *) ACPI_ADD_PTR (void, Resource,
278                   (Resource->Length - ItemCount * 2));
279             *(UINT16 **) Destination = ACPI_CAST_PTR (UINT16, Target);

281             /* Copy the PIN data */

283             Source = ACPI_ADD_PTR (void, Aml, ACPI_GET16 (Source));
284             AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
285             break;

287         case ACPI_RSC_MOVE_GPIO_RES:

289             /* Generate and set the ResourceSource string pointer */

291             Target = (char *) ACPI_ADD_PTR (void, Resource,
292                   (Resource->Length - ItemCount));
293             *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);

295             /* Copy the ResourceSource string */

297             Source = ACPI_ADD_PTR (void, Aml, ACPI_GET16 (Source));
298             AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
299             break;

301         case ACPI_RSC_MOVE_SERIAL_VEN:

303             /* Generate and set the Vendor Data pointer */

305             Target = (char *) ACPI_ADD_PTR (void, Resource,
306                   (Resource->Length - ItemCount));
307             *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);

309             /* Copy the Vendor Data */

311             Source = ACPI_ADD_PTR (void, Aml, Info->Value);
312             AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
313             break;

315         case ACPI_RSC_MOVE_SERIAL_RES:

317             /* Generate and set the ResourceSource string pointer */

319             Target = (char *) ACPI_ADD_PTR (void, Resource,
320                   (Resource->Length - ItemCount));
321             *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);

323             /* Copy the ResourceSource string */
```



```
325             Source = ACPI_ADD_PTR (void, Aml, (ACPI_GET16 (Source) + Info->Value
326             AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
327             break;

329         case ACPI_RSC_SET8:

331             ACPI_MEMSET (Destination, Info->AmlOffset, Info->Value);
332             break;

334         case ACPI_RSC_DATA8:

336             Target = ACPI_ADD_PTR (char, Resource, Info->Value);
337             ACPI_MEMCPY (Destination, Source,  ACPI_GET16 (Target));
338             break;

340         case ACPI_RSC_ADDRESS:
341             /*
342              * Common handler for address descriptor flags
343              */
344             if (!AcpiRsGetAddressCommon (Resource, Aml))
345             {
346                 return_ACPI_STATUS (AE_AML_INVALID_RESOURCE_TYPE);
347             }
348             break;

350         case ACPI_RSC_SOURCE:
351             /*
352              * Optional ResourceSource (Index and String)
353              */
354             Resource->Length +=
355                 AcpiRsGetResourceSource (AmlResourceLength, Info->Value,
356                     Destination, Aml, NULL);
357             break;

359         case ACPI_RSC_SOURCEX:
360             /*
361              * Optional ResourceSource (Index and String). This is the more
362              * complicated case used by the Interrupt() macro
363              */
364             Target = ACPI_ADD_PTR (char, Resource,
365                 Info->AmlOffset + (ItemCount * 4));

367             Resource->Length +=
368                 AcpiRsGetResourceSource (AmlResourceLength, (ACPI_RS_LENGTH)
369                     (((ItemCount - 1) * sizeof (UINT32)) + Info->Value),
370                     Destination, Aml, Target);
371             break;

373         case ACPI_RSC_BITMASK:
374             /*
375              * 8-bit encoded bitmask (DMA macro)
376              */
377             ItemCount = AcpiRsDecodeBitmask (ACPI_GET8 (Source), Destination);
378             if (ItemCount)
379             {
380                 Resource->Length += (ItemCount - 1);
381             }

383             Target = ACPI_ADD_PTR (char, Resource, Info->Value);
384             ACPI_SET8 (Target, ItemCount);
385             break;

387         case ACPI_RSC_BITMASK16:
388             /*
389              * 16-bit encoded bitmask (IRQ macro)
390              */
```

```
391             ACPI_MOVE_16_TO_16 (&Temp16, Source);

393             ItemCount = AcpiRsDecodeBitmask (Temp16, Destination);
394             if (ItemCount)
395             {
396                 Resource->Length += (ItemCount - 1);
397             }

399             Target = ACPI_ADD_PTR (char, Resource, Info->Value);
400             ACPI_SET8 (Target, ItemCount);
401             break;

403         case ACPI_RSC_EXIT_NE:
404             /*
405              * Control - Exit conversion if not equal
406              */
407             switch (Info->ResourceOffset)
408             {
409             case ACPI_RSC_COMPARE_AML_LENGTH:

411                 if (AmlResourceLength != Info->Value)
412                 {
413                     goto Exit;
414                 }
415                 break;

417             case ACPI_RSC_COMPARE_VALUE:

419                 if (ACPI_GET8 (Source) != Info->Value)
420                 {
421                     goto Exit;
422                 }
423                 break;

425             default:

427                 ACPI_ERROR ((AE_INFO, "Invalid conversion sub-opcode"));
428                 return_ACPI_STATUS (AE_BAD_PARAMETER);
429             }
430             break;

432         default:

434             ACPI_ERROR ((AE_INFO, "Invalid conversion opcode"));
435             return_ACPI_STATUS (AE_BAD_PARAMETER);
436         }

438         Count--;
439         Info++;
440     }

442 Exit:
443     if (!FlagsMode)
444     {
445         /* Round the resource struct length up to the next boundary (32 or 64) *

447         Resource->Length = (UINT32) ACPI_ROUND_UP_TO_NATIVE_WORD (Resource->Leng
448     }
449     return_ACPI_STATUS (AE_OK);
450 }


453 /*******************************************************************************
454  *
455  * FUNCTION:    AcpiRsConvertResourceToAml
456  *
```



```
457  * PARAMETERS:  Resource            - Pointer to the resource descriptor
458  *              Aml                 - Where the AML descriptor is returned
459  *              Info                - Pointer to appropriate conversion table
460  *
461  * RETURN:      Status
462  *
463  * DESCRIPTION: Convert an internal resource descriptor to the corresponding
464  *              external AML resource descriptor.
465  *
466  ******************************************************************************/

468 ACPI_STATUS
469 AcpiRsConvertResourceToAml (
470     ACPI_RESOURCE           *Resource,
471     AML_RESOURCE            *Aml,
472     ACPI_RSCONVERT_INFO     *Info)
473 {
474     void                    *Source = NULL;
475     void                    *Destination;
476     char                    *Target;
477     ACPI_RSDESC_SIZE        AmlLength = 0;
478     UINT8                   Count;
479     UINT16                  Temp16 = 0;
480     UINT16                  ItemCount = 0;


483     ACPI_FUNCTION_TRACE (RsConvertResourceToAml);


486     if (!Info)
487     {
488         return_ACPI_STATUS (AE_BAD_PARAMETER);
489     }

491     /*
492      * First table entry must be ACPI_RSC_INITxxx and must contain the
493      * table length (# of table entries)
494      */
495     Count = INIT_TABLE_LENGTH (Info);

497     while (Count)
498     {
499         /*
500          * Source is the internal resource descriptor,
501          * destination is the external AML byte stream buffer
502          */
503         Source      = ACPI_ADD_PTR (void, Resource, Info->ResourceOffset);
504         Destination = ACPI_ADD_PTR (void, Aml, Info->AmlOffset);

506         switch (Info->Opcode)
507         {
508         case ACPI_RSC_INITSET:

510             ACPI_MEMSET (Aml, 0, INIT_RESOURCE_LENGTH (Info));
511             AmlLength = INIT_RESOURCE_LENGTH (Info);
512             AcpiRsSetResourceHeader (INIT_RESOURCE_TYPE (Info), AmlLength, Aml);
513             break;

515         case ACPI_RSC_INITGET:
516             break;

518         case ACPI_RSC_FLAGINIT:
519             /*
520              * Clear the flag byte
521              */
522             ACPI_SET8 (Destination, 0);
```

```
523             break;
524
525         case ACPI_RSC_1BITFLAG:
526             /*
527              * Mask and shift the flag bit
528              */
529             ACPI_SET_BIT (*ACPI_CAST8 (Destination), (UINT8)
530                 ((ACPI_GET8 (Source) & 0x01) << Info->Value));
531             break;
532
533         case ACPI_RSC_2BITFLAG:
534             /*
535              * Mask and shift the flag bits
536              */
537             ACPI_SET_BIT (*ACPI_CAST8 (Destination), (UINT8)
538                 ((ACPI_GET8 (Source) & 0x03) << Info->Value));
539             break;
540
541         case ACPI_RSC_3BITFLAG:
542             /*
543              * Mask and shift the flag bits
544              */
545             ACPI_SET_BIT (*ACPI_CAST8 (Destination), (UINT8)
546                 ((ACPI_GET8 (Source) & 0x07) << Info->Value));
547             break;
548
549         case ACPI_RSC_COUNT:
550
551             ItemCount = ACPI_GET8 (Source);
552             ACPI_SET8 (Destination, ItemCount);
553
554             AmlLength = (UINT16) (AmlLength + (Info->Value * (ItemCount - 1)));
555             break;
556
557         case ACPI_RSC_COUNT16:
558
559             ItemCount = ACPI_GET16 (Source);
560             AmlLength = (UINT16) (AmlLength + ItemCount);
561             AcpiRsSetResourceLength (AmlLength, Aml);
562             break;
563
564         case ACPI_RSC_COUNT_GPIO_PIN:
565
566             ItemCount = ACPI_GET16 (Source);
567             ACPI_SET16 (Destination, AmlLength);
568
569             AmlLength = (UINT16) (AmlLength + ItemCount * 2);
570             Target = ACPI_ADD_PTR (void, Aml, Info->Value);
571             ACPI_SET16 (Target, AmlLength);
572             AcpiRsSetResourceLength (AmlLength, Aml);
573             break;
574
575         case ACPI_RSC_COUNT_GPIO_VEN:
576
577             ItemCount = ACPI_GET16 (Source);
578             ACPI_SET16 (Destination, ItemCount);
579
580             AmlLength = (UINT16) (AmlLength + (Info->Value * ItemCount));
581             AcpiRsSetResourceLength (AmlLength, Aml);
582             break;
583
584         case ACPI_RSC_COUNT_GPIO_RES:
585
586             /* Set resource source string length */
587
588             ItemCount = ACPI_GET16 (Source);
```



```
589             ACPI_SET16 (Destination, AmlLength);
590
591             /* Compute offset for the Vendor Data */
592
593             AmlLength = (UINT16) (AmlLength + ItemCount);
594             Target = ACPI_ADD_PTR (void, Aml, Info->Value);
595
596             /* Set vendor offset only if there is vendor data */
597
598             if (Resource->Data.Gpio.VendorLength)
599             {
600                 ACPI_SET16 (Target, AmlLength);
601             }
602
603             AcpiRsSetResourceLength (AmlLength, Aml);
604             break;
605
606         case ACPI_RSC_COUNT_SERIAL_VEN:
607
608             ItemCount = ACPI_GET16 (Source);
609             ACPI_SET16 (Destination, ItemCount + Info->Value);
610             AmlLength = (UINT16) (AmlLength + ItemCount);
611             AcpiRsSetResourceLength (AmlLength, Aml);
612             break;
613
614         case ACPI_RSC_COUNT_SERIAL_RES:
615
616             ItemCount = ACPI_GET16 (Source);
617             AmlLength = (UINT16) (AmlLength + ItemCount);
618             AcpiRsSetResourceLength (AmlLength, Aml);
619             break;
620
621         case ACPI_RSC_LENGTH:
622
623             AcpiRsSetResourceLength (Info->Value, Aml);
624             break;
625
626         case ACPI_RSC_MOVE8:
627         case ACPI_RSC_MOVE16:
628         case ACPI_RSC_MOVE32:
629         case ACPI_RSC_MOVE64:
630
631             if (Info->Value)
632             {
633                 ItemCount = Info->Value;
634             }
635             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
636             break;
637
638         case ACPI_RSC_MOVE_GPIO_PIN:
639
640             Destination = (char *) ACPI_ADD_PTR (void, Aml,
641                   ACPI_GET16 (Destination));
642             Source = * (UINT16 **) Source;
643             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
644             break;
645
646         case ACPI_RSC_MOVE_GPIO_RES:
647
648             /* Used for both ResourceSource string and VendorData */
649
650             Destination = (char *) ACPI_ADD_PTR (void, Aml,
651                   ACPI_GET16 (Destination));
652             Source = * (UINT8 **) Source;
653             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
654             break;
```

```
656         case ACPI_RSC_MOVE_SERIAL_VEN:

658             Destination = (char *) ACPI_ADD_PTR (void, Aml,
659                   (AmlLength - ItemCount));
660             Source = * (UINT8 **) Source;
661             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
662             break;

664         case ACPI_RSC_MOVE_SERIAL_RES:

666             Destination = (char *) ACPI_ADD_PTR (void, Aml,
667                   (AmlLength - ItemCount));
668             Source = * (UINT8 **) Source;
669             AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
670             break;

672         case ACPI_RSC_ADDRESS:

674             /* Set the Resource Type, General Flags, and Type-Specific Flags */

676             AcpiRsSetAddressCommon (Aml, Resource);
677             break;

679         case ACPI_RSC_SOURCEX:
680             /*
681              * Optional ResourceSource (Index and String)
682              */
683             AmlLength = AcpiRsSetResourceSource (
684                             Aml, (ACPI_RS_LENGTH) AmlLength, Source);
685             AcpiRsSetResourceLength (AmlLength, Aml);
686             break;

688         case ACPI_RSC_SOURCE:
689             /*
690              * Optional ResourceSource (Index and String). This is the more
691              * complicated case used by the Interrupt() macro
692              */
693             AmlLength = AcpiRsSetResourceSource (Aml, Info->Value, Source);
694             AcpiRsSetResourceLength (AmlLength, Aml);
695             break;

697         case ACPI_RSC_BITMASK:
698             /*
699              * 8-bit encoded bitmask (DMA macro)
700              */
701             ACPI_SET8 (Destination,
702                 AcpiRsEncodeBitmask (Source,
703                     *ACPI_ADD_PTR (UINT8, Resource, Info->Value)));
704             break;

706         case ACPI_RSC_BITMASK16:
707             /*
708              * 16-bit encoded bitmask (IRQ macro)
709              */
710             Temp16 = AcpiRsEncodeBitmask (Source,
711                         *ACPI_ADD_PTR (UINT8, Resource, Info->Value));
712             ACPI_MOVE_16_TO_16 (Destination, &Temp16);
713             break;

715         case ACPI_RSC_EXIT_LE:
716             /*
717              * Control - Exit conversion if less than or equal
718              */
719             if (ItemCount <= Info->Value)
720             {
```



```
721                 goto Exit;
722             }
723             break;

725         case ACPI_RSC_EXIT_NE:
726             /*
727              * Control - Exit conversion if not equal
728              */
729             switch (COMPARE_OPCODE (Info))
730             {
731             case ACPI_RSC_COMPARE_VALUE:

733                 if (*ACPI_ADD_PTR (UINT8, Resource,
734                         COMPARE_TARGET (Info)) != COMPARE_VALUE (Info))
735                 {
736                     goto Exit;
737                 }
738                 break;

740             default:

742                 ACPI_ERROR ((AE_INFO, "Invalid conversion sub-opcode"));
743                 return_ACPI_STATUS (AE_BAD_PARAMETER);
744             }
745             break;

747         case ACPI_RSC_EXIT_EQ:
748             /*
749              * Control - Exit conversion if equal
750              */
751             if (*ACPI_ADD_PTR (UINT8, Resource,
752                     COMPARE_TARGET (Info)) == COMPARE_VALUE (Info))
753             {
754                 goto Exit;
755             }
756             break;

758         default:

760             ACPI_ERROR ((AE_INFO, "Invalid conversion opcode"));
761             return_ACPI_STATUS (AE_BAD_PARAMETER);
762         }

764         Count--;
765         Info++;
766     }

768 Exit:
769     return_ACPI_STATUS (AE_OK);
770 }


773 #if 0
774 /* Previous resource validations */

776     if (Aml->ExtAddress64.RevisionID != AML_RESOURCE_EXTENDED_ADDRESS_REVISION)
777     {
778         return_ACPI_STATUS (AE_SUPPORT);
779     }

781     if (Resource->Data.StartDpf.PerformanceRobustness >= 3)
782     {
783         return_ACPI_STATUS (AE_AML_BAD_RESOURCE_VALUE);
784     }

786     if (((Aml->Irq.Flags & 0x09) == 0x00) ||
```

```
787         ((Aml->Irq.Flags & 0x09) == 0x09))
788     {
789         /*
790          * Only [ActiveHigh, EdgeSensitive] or [ActiveLow, LevelSensitive]
791          * polarity/trigger interrupts are allowed (ACPI spec, section
792          * "IRQ Format"), so 0x00 and 0x09 are illegal.
793          */
794         ACPI_ERROR ((AE_INFO,
795             "Invalid interrupt polarity/trigger in resource list, 0x%X",
796             Aml->Irq.Flags));
797         return_ACPI_STATUS (AE_BAD_DATA);
798     }

800     Resource->Data.ExtendedIrq.InterruptCount = Temp8;
801     if (Temp8 < 1)
802     {
803         /* Must have at least one IRQ */

805         return_ACPI_STATUS (AE_AML_BAD_RESOURCE_LENGTH);
806     }

808     if (Resource->Data.Dma.Transfer == 0x03)
809     {
810         ACPI_ERROR ((AE_INFO,
811             "Invalid DMA.Transfer preference (3)"));
812         return_ACPI_STATUS (AE_BAD_DATA);
813     }
814 #endif
```

```
**********************************************************
   17026 Thu Dec 26 13:49:31 2013
new/usr/src/common/acpica/components/resources/rsserial.c
acpica-unix2-20130823
**********************************************************
```

```
/*******************************************************************************
 *
 * Module Name: rsserial - GPIO/SerialBus resource descriptors
 *
 ******************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __RSIRQ_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"

#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsserial")


/*******************************************************************************
 *
 * AcpiRsConvertGpio
 *
 ******************************************************************************/

ACPI_RSCONVERT_INFO     AcpiRsConvertGpio[18] =
{
```



```
    {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_GPIO,
                        ACPI_RS_SIZE (ACPI_RESOURCE_GPIO),
                        ACPI_RSC_TABLE_SIZE (AcpiRsConvertGpio)},

    {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_GPIO,
                        sizeof (AML_RESOURCE_GPIO),
                        0},

    /*
     * These fields are contiguous in both the source and destination:
     * RevisionId
     * ConnectionType
     */
    {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.Gpio.RevisionId),
                        AML_OFFSET (Gpio.RevisionId),
                        2},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.ProducerConsumer),
                        AML_OFFSET (Gpio.Flags),
                        0},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Sharable),
                        AML_OFFSET (Gpio.IntFlags),
                        3},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.WakeCapable),
                        AML_OFFSET (Gpio.IntFlags),
                        4},

    {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.Gpio.IoRestriction),
                        AML_OFFSET (Gpio.IntFlags),
                        0},

    {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Triggering),
                        AML_OFFSET (Gpio.IntFlags),
                        0},

    {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Polarity),
                        AML_OFFSET (Gpio.IntFlags),
                        1},

    {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.Gpio.PinConfig),
                        AML_OFFSET (Gpio.PinConfig),
                        1},

    /*
     * These fields are contiguous in both the source and destination:
     * DriveStrength
     * DebounceTimeout
     */
    {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.Gpio.DriveStrength),
                        AML_OFFSET (Gpio.DriveStrength),
                        2},

    /* Pin Table */

    {ACPI_RSC_COUNT_GPIO_PIN, ACPI_RS_OFFSET (Data.Gpio.PinTableLength),
                        AML_OFFSET (Gpio.PinTableOffset),
                        AML_OFFSET (Gpio.ResSourceOffset)},

    {ACPI_RSC_MOVE_GPIO_PIN, ACPI_RS_OFFSET (Data.Gpio.PinTable),
                        AML_OFFSET (Gpio.PinTableOffset),
                        0},

    /* Resource Source */
```

```
128     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.Gpio.ResourceSource.Index),
129                         AML_OFFSET (Gpio.ResSourceIndex),
130                         1},

132     {ACPI_RSC_COUNT_GPIO_RES,  ACPI_RS_OFFSET (Data.Gpio.ResourceSource.StringLe
133                         AML_OFFSET (Gpio.ResSourceOffset),
134                         AML_OFFSET (Gpio.VendorOffset)},

136     {ACPI_RSC_MOVE_GPIO_RES,   ACPI_RS_OFFSET (Data.Gpio.ResourceSource.StringPt
137                         AML_OFFSET (Gpio.ResSourceOffset),
138                         0},

140     /* Vendor Data */

142     {ACPI_RSC_COUNT_GPIO_VEN,   ACPI_RS_OFFSET (Data.Gpio.VendorLength),
143                         AML_OFFSET (Gpio.VendorLength),
144                         1},

146     {ACPI_RSC_MOVE_GPIO_RES,   ACPI_RS_OFFSET (Data.Gpio.VendorData),
147                         AML_OFFSET (Gpio.VendorOffset),
148                         0},
149 };


152 /*******************************************************************************
153  *
154  * AcpiRsConvertI2cSerialBus
155  *
156  ******************************************************************************/

158 ACPI_RSCONVERT_INFO     AcpiRsConvertI2cSerialBus[16] =
159 {
160     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_SERIAL_BUS,
161                         ACPI_RS_SIZE (ACPI_RESOURCE_I2C_SERIALBUS),
162                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertI2cSerialBus)},

164     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_SERIAL_BUS,
165                         sizeof (AML_RESOURCE_I2C_SERIALBUS),
166                         0},

168     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.RevisionId),
169                         AML_OFFSET (CommonSerialBus.RevisionId),
170                         1},

172     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.Type),
173                         AML_OFFSET (CommonSerialBus.Type),
174                         1},

176     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.SlaveMode),
177                         AML_OFFSET (CommonSerialBus.Flags),
178                         0},

180     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.ProducerConsumer),
181                         AML_OFFSET (CommonSerialBus.Flags),
182                         1},

184     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.TypeRevisionId),
185                         AML_OFFSET (CommonSerialBus.TypeRevisionId),
186                         1},

188     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.CommonSerialBus.TypeDataLength),
189                         AML_OFFSET (CommonSerialBus.TypeDataLength),
190                         1},

192     /* Vendor data */
```



```
194     {ACPI_RSC_COUNT_SERIAL_VEN, ACPI_RS_OFFSET (Data.CommonSerialBus.VendorLengt
195                         AML_OFFSET (CommonSerialBus.TypeDataLength),
196                         AML_RESOURCE_I2C_MIN_DATA_LEN},

198     {ACPI_RSC_MOVE_SERIAL_VEN,  ACPI_RS_OFFSET (Data.CommonSerialBus.VendorData)
199                         0,
200                         sizeof (AML_RESOURCE_I2C_SERIALBUS)},

202     /* Resource Source */

204     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSource.Inde
205                         AML_OFFSET (CommonSerialBus.ResSourceIndex),
206                         1},

208     {ACPI_RSC_COUNT_SERIAL_RES, ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
209                         AML_OFFSET (CommonSerialBus.TypeDataLength),
210                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},

212     {ACPI_RSC_MOVE_SERIAL_RES,  ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
213                         AML_OFFSET (CommonSerialBus.TypeDataLength),
214                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},

216     /* I2C bus type specific */

218     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.I2cSerialBus.AccessMode),
219                         AML_OFFSET (I2cSerialBus.TypeSpecificFlags),
220                         0},

222     {ACPI_RSC_MOVE32,   ACPI_RS_OFFSET (Data.I2cSerialBus.ConnectionSpeed),
223                         AML_OFFSET (I2cSerialBus.ConnectionSpeed),
224                         1},

226     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.I2cSerialBus.SlaveAddress),
227                         AML_OFFSET (I2cSerialBus.SlaveAddress),
228                         1},
229 };


232 /*******************************************************************************
233  *
234  * AcpiRsConvertSpiSerialBus
235  *
236  ******************************************************************************/

238 ACPI_RSCONVERT_INFO     AcpiRsConvertSpiSerialBus[20] =
239 {
240     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_SERIAL_BUS,
241                         ACPI_RS_SIZE (ACPI_RESOURCE_SPI_SERIALBUS),
242                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertSpiSerialBus)},

244     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_SERIAL_BUS,
245                         sizeof (AML_RESOURCE_SPI_SERIALBUS),
246                         0},

248     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.RevisionId),
249                         AML_OFFSET (CommonSerialBus.RevisionId),
250                         1},

252     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.Type),
253                         AML_OFFSET (CommonSerialBus.Type),
254                         1},

256     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.SlaveMode),
257                         AML_OFFSET (CommonSerialBus.Flags),
258                         0},
```

```
260     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.ProducerConsumer),
261                         AML_OFFSET (CommonSerialBus.Flags),
262                         1},

264     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.TypeRevisionId),
265                         AML_OFFSET (CommonSerialBus.TypeRevisionId),
266                         1},

268     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.CommonSerialBus.TypeDataLength),
269                         AML_OFFSET (CommonSerialBus.TypeDataLength),
270                         1},

272     /* Vendor data */

274     {ACPI_RSC_COUNT_SERIAL_VEN, ACPI_RS_OFFSET (Data.CommonSerialBus.VendorLengt
275                         AML_OFFSET (CommonSerialBus.TypeDataLength),
276                         AML_RESOURCE_SPI_MIN_DATA_LEN},

278     {ACPI_RSC_MOVE_SERIAL_VEN,  ACPI_RS_OFFSET (Data.CommonSerialBus.VendorData)
279                         0,
280                         sizeof (AML_RESOURCE_SPI_SERIALBUS)},

282     /* Resource Source */

284     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSource.Inde
285                         AML_OFFSET (CommonSerialBus.ResSourceIndex),
286                         1},

288     {ACPI_RSC_COUNT_SERIAL_RES, ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
289                         AML_OFFSET (CommonSerialBus.TypeDataLength),
290                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},

292     {ACPI_RSC_MOVE_SERIAL_RES,  ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
293                         AML_OFFSET (CommonSerialBus.TypeDataLength),
294                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},

296     /* Spi bus type specific  */

298     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.SpiSerialBus.WireMode),
299                         AML_OFFSET (SpiSerialBus.TypeSpecificFlags),
300                         0},

302     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.SpiSerialBus.DevicePolarity),
303                         AML_OFFSET (SpiSerialBus.TypeSpecificFlags),
304                         1},

306     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.SpiSerialBus.DataBitLength),
307                         AML_OFFSET (SpiSerialBus.DataBitLength),
308                         1},

310     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.SpiSerialBus.ClockPhase),
311                         AML_OFFSET (SpiSerialBus.ClockPhase),
312                         1},

314     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.SpiSerialBus.ClockPolarity),
315                         AML_OFFSET (SpiSerialBus.ClockPolarity),
316                         1},

318     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.SpiSerialBus.DeviceSelection),
319                         AML_OFFSET (SpiSerialBus.DeviceSelection),
320                         1},

322     {ACPI_RSC_MOVE32,   ACPI_RS_OFFSET (Data.SpiSerialBus.ConnectionSpeed),
323                         AML_OFFSET (SpiSerialBus.ConnectionSpeed),
324                         1},
325 };
```



```
328 /*******************************************************************************
329  *
330  * AcpiRsConvertUartSerialBus
331  *
332  ******************************************************************************/

334 ACPI_RSCONVERT_INFO     AcpiRsConvertUartSerialBus[22] =
335 {
336     {ACPI_RSC_INITGET,  ACPI_RESOURCE_TYPE_SERIAL_BUS,
337                         ACPI_RS_SIZE (ACPI_RESOURCE_UART_SERIALBUS),
338                         ACPI_RSC_TABLE_SIZE (AcpiRsConvertUartSerialBus)},

340     {ACPI_RSC_INITSET,  ACPI_RESOURCE_NAME_SERIAL_BUS,
341                         sizeof (AML_RESOURCE_UART_SERIALBUS),
342                         0},

344     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.RevisionId),
345                         AML_OFFSET (CommonSerialBus.RevisionId),
346                         1},

348     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.Type),
349                         AML_OFFSET (CommonSerialBus.Type),
350                         1},

352     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.SlaveMode),
353                         AML_OFFSET (CommonSerialBus.Flags),
354                         0},

356     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.CommonSerialBus.ProducerConsumer),
357                         AML_OFFSET (CommonSerialBus.Flags),
358                         1},

360     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.TypeRevisionId),
361                         AML_OFFSET (CommonSerialBus.TypeRevisionId),
362                         1},

364     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.CommonSerialBus.TypeDataLength),
365                         AML_OFFSET (CommonSerialBus.TypeDataLength),
366                         1},

368     /* Vendor data */

370     {ACPI_RSC_COUNT_SERIAL_VEN, ACPI_RS_OFFSET (Data.CommonSerialBus.VendorLengt
371                         AML_OFFSET (CommonSerialBus.TypeDataLength),
372                         AML_RESOURCE_UART_MIN_DATA_LEN},

374     {ACPI_RSC_MOVE_SERIAL_VEN,  ACPI_RS_OFFSET (Data.CommonSerialBus.VendorData)
375                         0,
376                         sizeof (AML_RESOURCE_UART_SERIALBUS)},

378     /* Resource Source */

380     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSource.Inde
381                         AML_OFFSET (CommonSerialBus.ResSourceIndex),
382                         1},

384     {ACPI_RSC_COUNT_SERIAL_RES, ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
385                         AML_OFFSET (CommonSerialBus.TypeDataLength),
386                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},

388     {ACPI_RSC_MOVE_SERIAL_RES,  ACPI_RS_OFFSET (Data.CommonSerialBus.ResourceSou
389                         AML_OFFSET (CommonSerialBus.TypeDataLength),
390                         sizeof (AML_RESOURCE_COMMON_SERIALBUS)},
```

```
392     /* Uart bus type specific  */
393
394     {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.FlowControl),
395                         AML_OFFSET (UartSerialBus.TypeSpecificFlags),
396                         0},
397
398     {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.StopBits),
399                         AML_OFFSET (UartSerialBus.TypeSpecificFlags),
400                         2},
401
402     {ACPI_RSC_3BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.DataBits),
403                         AML_OFFSET (UartSerialBus.TypeSpecificFlags),
404                         4},
405
406     {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.Endian),
407                         AML_OFFSET (UartSerialBus.TypeSpecificFlags),
408                         7},
409
410     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.UartSerialBus.Parity),
411                         AML_OFFSET (UartSerialBus.Parity),
412                         1},
413
414     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.UartSerialBus.LinesEnabled),
415                         AML_OFFSET (UartSerialBus.LinesEnabled),
416                         1},
417
418     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.UartSerialBus.RxFifoSize),
419                         AML_OFFSET (UartSerialBus.RxFifoSize),
420                         1},
421
422     {ACPI_RSC_MOVE16,   ACPI_RS_OFFSET (Data.UartSerialBus.TxFifoSize),
423                         AML_OFFSET (UartSerialBus.TxFifoSize),
424                         1},
425
426     {ACPI_RSC_MOVE32,   ACPI_RS_OFFSET (Data.UartSerialBus.DefaultBaudRate),
427                         AML_OFFSET (UartSerialBus.DefaultBaudRate),
428                         1},
429 };
```

```
**********************************************************
   25635 Thu Dec 26 13:49:32 2013
new/usr/src/common/acpica/components/resources/rsutils.c
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: rsutils - Utilities for the resource manager
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __RSUTILS_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acresrc.h"


#define _COMPONENT          ACPI_RESOURCES
        ACPI_MODULE_NAME    ("rsutils")


/******************************************************************************
 *
```



```
 * FUNCTION:    AcpiRsDecodeBitmask
 *
 * PARAMETERS:  Mask            - Bitmask to decode
 *              List            - Where the converted list is returned
 *
 * RETURN:      Count of bits set (length of list)
 *
 * DESCRIPTION: Convert a bit mask into a list of values
 *
 ******************************************************************************/

UINT8
AcpiRsDecodeBitmask (
    UINT16                  Mask,
    UINT8                   *List)
{
    UINT8                   i;
    UINT8                   BitCount;


    ACPI_FUNCTION_ENTRY ();


    /* Decode the mask bits */

    for (i = 0, BitCount = 0; Mask; i++)
    {
        if (Mask & 0x0001)
        {
            List[BitCount] = i;
            BitCount++;
        }

        Mask >>= 1;
    }

    return (BitCount);
}
```

*unchanged_portion_omitted_*

```
/******************************************************************************
 *
 * FUNCTION:    AcpiRsMoveData
 *
 * PARAMETERS:  Destination         - Pointer to the destination descriptor
 *              Source              - Pointer to the source descriptor
 *              ItemCount           - How many items to move
 *              MoveType            - Byte width
 *
 * RETURN:      None
 *
 * DESCRIPTION: Move multiple data items from one descriptor to another. Handles
 *              alignment issues and endian issues if necessary, as configured
 *              via the ACPI_MOVE_* macros. (This is why a memcpy is not used)
 *
 *****************************************************************************/

void
AcpiRsMoveData (
    void                    *Destination,
    void                    *Source,
    UINT16                  ItemCount,
    UINT8                   MoveType)
{
    UINT32                  i;
```

```
162     ACPI_FUNCTION_ENTRY ();


165     /* One move per item */

167     for (i = 0; i < ItemCount; i++)
168     {
169         switch (MoveType)
170         {
171         /*
172          * For the 8-bit case, we can perform the move all at once
173          * since there are no alignment or endian issues
174          */
175         case ACPI_RSC_MOVE8:
176         case ACPI_RSC_MOVE_GPIO_RES:
177         case ACPI_RSC_MOVE_SERIAL_VEN:
178         case ACPI_RSC_MOVE_SERIAL_RES:

180             ACPI_MEMCPY (Destination, Source, ItemCount);
181             return;

183         /*
184          * 16-, 32-, and 64-bit cases must use the move macros that perform
185          * endian conversion and/or accommodate hardware that cannot perform
181          * endian conversion and/or accomodate hardware that cannot perform
186          * misaligned memory transfers
187          */
188         case ACPI_RSC_MOVE16:
189         case ACPI_RSC_MOVE_GPIO_PIN:

191             ACPI_MOVE_16_TO_16 (&ACPI_CAST_PTR (UINT16, Destination)[i],
192                                 &ACPI_CAST_PTR (UINT16, Source)[i]);
193             break;

195         case ACPI_RSC_MOVE32:

197             ACPI_MOVE_32_TO_32 (&ACPI_CAST_PTR (UINT32, Destination)[i],
198                                 &ACPI_CAST_PTR (UINT32, Source)[i]);
199             break;

201         case ACPI_RSC_MOVE64:

203             ACPI_MOVE_64_TO_64 (&ACPI_CAST_PTR (UINT64, Destination)[i],
204                                 &ACPI_CAST_PTR (UINT64, Source)[i]);
205             break;

207         default:

209             return;
210         }
211     }
212 }
```

*_____unchanged_portion_omitted_*

```
663 /******************************************************************************
664  *
665  * FUNCTION:    AcpiRsGetAeiMethodData
666  *
667  * PARAMETERS:  Node            - Device node
668  *              RetBuffer       - Pointer to a buffer structure for the
669  *                                results
670  *
671  * RETURN:      Status
```



```
672  *
673  * DESCRIPTION: This function is called to get the _AEI value of an object
674  *              contained in an object specified by the handle passed in
675  *
676  *              If the function fails an appropriate status will be returned
677  *              and the contents of the callers buffer is undefined.
678  *
679  *****************************************************************************/

681 ACPI_STATUS
682 AcpiRsGetAeiMethodData (
683     ACPI_NAMESPACE_NODE     *Node,
684     ACPI_BUFFER             *RetBuffer)
685 {
686     ACPI_OPERAND_OBJECT     *ObjDesc;
687     ACPI_STATUS             Status;


690     ACPI_FUNCTION_TRACE (RsGetAeiMethodData);


693     /* Parameters guaranteed valid by caller */

695     /* Execute the method, no parameters */

697     Status = AcpiUtEvaluateObject (Node, METHOD_NAME__AEI,
698                 ACPI_BTYPE_BUFFER, &ObjDesc);
699     if (ACPI_FAILURE (Status))
700     {
701         return_ACPI_STATUS (Status);
702     }

704     /*
705      * Make the call to create a resource linked list from the
706      * byte stream buffer that comes back from the _CRS method
707      * execution.
708      */
709     Status = AcpiRsCreateResourceList (ObjDesc, RetBuffer);

711     /* On exit, we must delete the object returned by evaluateObject */

713     AcpiUtRemoveReference (ObjDesc);
714     return_ACPI_STATUS (Status);
715 }


718 /******************************************************************************
719  *
720  * FUNCTION:    AcpiRsGetMethodData
721  *
722  * PARAMETERS:  Handle          - Handle to the containing object
723  *              Path            - Path to method, relative to Handle
724  *              RetBuffer       - Pointer to a buffer structure for the
725  *                                results
726  *
727  * RETURN:      Status
728  *
729  * DESCRIPTION: This function is called to get the _CRS or _PRS value of an
730  *              object contained in an object specified by the handle passed in
731  *
732  *              If the function fails an appropriate status will be returned
733  *              and the contents of the callers buffer is undefined.
734  *
735  *****************************************************************************/

737 ACPI_STATUS
```

```
 738 AcpiRsGetMethodData (
 739     ACPI_HANDLE             Handle,
 740     char                    *Path,
 741     ACPI_BUFFER             *RetBuffer)
 742 {
 743     ACPI_OPERAND_OBJECT     *ObjDesc;
 744     ACPI_STATUS             Status;


 747     ACPI_FUNCTION_TRACE (RsGetMethodData);


 750     /* Parameters guaranteed valid by caller */

 752     /* Execute the method, no parameters */

 754     Status = AcpiUtEvaluateObject (ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Handle),
 755         Path, ACPI_BTYPE_BUFFER, &ObjDesc);
 690     Status = AcpiUtEvaluateObject (Handle, Path, ACPI_BTYPE_BUFFER, &ObjDesc);
 756     if (ACPI_FAILURE (Status))
 757     {
 758         return_ACPI_STATUS (Status);
 759     }

 761     /*
 762      * Make the call to create a resource linked list from the
 763      * byte stream buffer that comes back from the method
 764      * execution.
 765      */
 766     Status = AcpiRsCreateResourceList (ObjDesc, RetBuffer);

 768     /* On exit, we must delete the object returned by EvaluateObject */

 770     AcpiUtRemoveReference (ObjDesc);
 771     return_ACPI_STATUS (Status);
 772 }


 775 /*******************************************************************************
 776  *
 777  * FUNCTION:    AcpiRsSetSrsMethodData
 778  *
 779  * PARAMETERS:  Node            - Device node
 780  *              InBuffer        - Pointer to a buffer structure of the
 781  *                                parameter
 782  *
 783  * RETURN:      Status
 784  *
 785  * DESCRIPTION: This function is called to set the _SRS of an object contained
 786  *              in an object specified by the handle passed in
 787  *
 788  *              If the function fails an appropriate status will be returned
 789  *              and the contents of the callers buffer is undefined.
 790  *
 791  * Note: Parameters guaranteed valid by caller
 792  *
 793  ******************************************************************************/

 795 ACPI_STATUS
 796 AcpiRsSetSrsMethodData (
 797     ACPI_NAMESPACE_NODE     *Node,
 798     ACPI_BUFFER             *InBuffer)
 799 {
 800     ACPI_EVALUATE_INFO      *Info;
 801     ACPI_OPERAND_OBJECT     *Args[2];
 802     ACPI_STATUS             Status;
```



```
 803     ACPI_BUFFER             Buffer;


 806     ACPI_FUNCTION_TRACE (RsSetSrsMethodData);


 809     /* Allocate and initialize the evaluation information block */

 811     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
 812     if (!Info)
 813     {
 814         return_ACPI_STATUS (AE_NO_MEMORY);
 815     }

 817     Info->PrefixNode = Node;
 818     Info->RelativePathname = METHOD_NAME__SRS;
 753     Info->Pathname = METHOD_NAME__SRS;
 819     Info->Parameters = Args;
 820     Info->Flags = ACPI_IGNORE_RETURN_VALUE;

 822     /*
 823      * The InBuffer parameter will point to a linked list of
 824      * resource parameters. It needs to be formatted into a
 825      * byte stream to be sent in as an input parameter to _SRS
 826      *
 827      * Convert the linked list into a byte stream
 828      */
 829     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
 830     Status = AcpiRsCreateAmlResources (InBuffer, &Buffer);
 765     Status = AcpiRsCreateAmlResources (InBuffer->Pointer, &Buffer);
 831     if (ACPI_FAILURE (Status))
 832     {
 833         goto Cleanup;
 834     }

 836     /* Create and initialize the method parameter object */

 838     Args[0] = AcpiUtCreateInternalObject (ACPI_TYPE_BUFFER);
 839     if (!Args[0])
 840     {
 841         /*
 842          * Must free the buffer allocated above (otherwise it is freed
 843          * later)
 844          */
 845         ACPI_FREE (Buffer.Pointer);
 846         Status = AE_NO_MEMORY;
 847         goto Cleanup;
 848     }

 850     Args[0]->Buffer.Length  = (UINT32) Buffer.Length;
 851     Args[0]->Buffer.Pointer = Buffer.Pointer;
 852     Args[0]->Common.Flags   = AOPOBJ_DATA_VALID;
 853     Args[1] = NULL;

 855     /* Execute the method, no return value is expected */

 857     Status = AcpiNsEvaluate (Info);

 859     /* Clean up and return the status from AcpiNsEvaluate */

 861     AcpiUtRemoveReference (Args[0]);

 863 Cleanup:
 864     ACPI_FREE (Info);
 865     return_ACPI_STATUS (Status);
 866 }
```

```
**********************************************************
   22711 Thu Dec 26 13:49:32 2013
new/usr/src/common/acpica/components/resources/rsxface.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: rsxface - Public interfaces to the resource manager
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __RSXFACE_C__
  46 #define EXPORT_ACPI_INTERFACES

  48 #include "acpi.h"
  49 #include "accommon.h"
  50 #include "acresrc.h"
  51 #include "acnamesp.h"

  53 #define _COMPONENT          ACPI_RESOURCES
  54         ACPI_MODULE_NAME    ("rsxface")

  56 /* Local macros for 16,32-bit to 64-bit conversion */

  58 #define ACPI_COPY_FIELD(Out, In, Field)  ((Out)->Field = (In)->Field)
```



```
  59 #define ACPI_COPY_ADDRESS(Out, In)                      \
  60     ACPI_COPY_FIELD(Out, In, ResourceType);              \
  61     ACPI_COPY_FIELD(Out, In, ProducerConsumer);          \
  62     ACPI_COPY_FIELD(Out, In, Decode);                    \
  63     ACPI_COPY_FIELD(Out, In, MinAddressFixed);           \
  64     ACPI_COPY_FIELD(Out, In, MaxAddressFixed);           \
  65     ACPI_COPY_FIELD(Out, In, Info);                      \
  66     ACPI_COPY_FIELD(Out, In, Granularity);               \
  67     ACPI_COPY_FIELD(Out, In, Minimum);                   \
  68     ACPI_COPY_FIELD(Out, In, Maximum);                   \
  69     ACPI_COPY_FIELD(Out, In, TranslationOffset);         \
  70     ACPI_COPY_FIELD(Out, In, AddressLength);             \
  71     ACPI_COPY_FIELD(Out, In, ResourceSource);


  74 /* Local prototypes */

  76 static ACPI_STATUS
  77 AcpiRsMatchVendorResource (
  78     ACPI_RESOURCE           *Resource,
  79     void                    *Context);

  81 static ACPI_STATUS
  82 AcpiRsValidateParameters (
  83     ACPI_HANDLE             DeviceHandle,
  84     ACPI_BUFFER             *Buffer,
  85     ACPI_NAMESPACE_NODE     **ReturnNode);


  88 /******************************************************************************
  89  *
  90  * FUNCTION:    AcpiRsValidateParameters
  91  *
  92  * PARAMETERS:  DeviceHandle    - Handle to a device
  93  *              Buffer          - Pointer to a data buffer
  94  *              ReturnNode      - Pointer to where the device node is returned
  95  *
  96  * RETURN:      Status
  97  *
  98  * DESCRIPTION: Common parameter validation for resource interfaces
  99  *
 100  *****************************************************************************/

 102 static ACPI_STATUS
 103 AcpiRsValidateParameters (
 104     ACPI_HANDLE             DeviceHandle,
 105     ACPI_BUFFER             *Buffer,
 106     ACPI_NAMESPACE_NODE     **ReturnNode)
 107 {
 108     ACPI_STATUS             Status;
 109     ACPI_NAMESPACE_NODE     *Node;


 112     ACPI_FUNCTION_TRACE (RsValidateParameters);


 115     /*
 116      * Must have a valid handle to an ACPI device
 117      */
 118     if (!DeviceHandle)
 119     {
 120         return_ACPI_STATUS (AE_BAD_PARAMETER);
 121     }

 123     Node = AcpiNsValidateHandle (DeviceHandle);
 124     if (!Node)
```

```
125     {
126         return_ACPI_STATUS (AE_BAD_PARAMETER);
127     }

129     if (Node->Type != ACPI_TYPE_DEVICE)
130     {
131         return_ACPI_STATUS (AE_TYPE);
132     }

134     /*
135      * Validate the user buffer object
136      *
137      * if there is a non-zero buffer length we also need a valid pointer in
138      * the buffer. If it's a zero buffer length, we'll be returning the
139      * needed buffer size (later), so keep going.
140      */
141     Status = AcpiUtValidateBuffer (Buffer);
142     if (ACPI_FAILURE (Status))
143     {
144         return_ACPI_STATUS (Status);
145     }

147     *ReturnNode = Node;
148     return_ACPI_STATUS (AE_OK);
149 }
_____unchanged_portion_omitted_

352 ACPI_EXPORT_SYMBOL (AcpiSetCurrentResources)


355 /*******************************************************************************
356  *
357  * FUNCTION:    AcpiGetEventResources
358  *
359  * PARAMETERS:  DeviceHandle    - Handle to the device object for the
360  *                                device we are getting resources
361  *              InBuffer        - Pointer to a buffer containing the
362  *                                resources to be set for the device
363  *
364  * RETURN:      Status
365  *
366  * DESCRIPTION: This function is called to get the event resources for a
367  *              specific device. The caller must first acquire a handle for
368  *              the desired device. The resource data is passed to the routine
369  *              the buffer pointed to by the InBuffer variable. Uses the
370  *              _AEI method.
371  *
372  ******************************************************************************/

374 ACPI_STATUS
375 AcpiGetEventResources (
376     ACPI_HANDLE             DeviceHandle,
377     ACPI_BUFFER             *RetBuffer)
378 {
379     ACPI_STATUS             Status;
380     ACPI_NAMESPACE_NODE     *Node;


383     ACPI_FUNCTION_TRACE (AcpiGetEventResources);


386     /* Validate parameters then dispatch to internal routine */

388     Status = AcpiRsValidateParameters (DeviceHandle, RetBuffer, &Node);
389     if (ACPI_FAILURE (Status))
390     {
```



```
391         return_ACPI_STATUS (Status);
392     }

394     Status = AcpiRsGetAeiMethodData (Node, RetBuffer);
395     return_ACPI_STATUS (Status);
396 }

398 ACPI_EXPORT_SYMBOL (AcpiGetEventResources)


401 /*******************************************************************************
402  *
403  * FUNCTION:    AcpiResourceToAddress64
404  *
405  * PARAMETERS:  Resource        - Pointer to a resource
406  *              Out             - Pointer to the users's return buffer
407  *                                (a struct acpi_resource_address64)
408  *
409  * RETURN:      Status
410  *
411  * DESCRIPTION: If the resource is an address16, address32, or address64,
412  *              copy it to the address64 return buffer. This saves the
413  *              caller from having to duplicate code for different-sized
414  *              addresses.
415  *
416  ******************************************************************************/

418 ACPI_STATUS
419 AcpiResourceToAddress64 (
420     ACPI_RESOURCE               *Resource,
421     ACPI_RESOURCE_ADDRESS64     *Out)
422 {
423     ACPI_RESOURCE_ADDRESS16     *Address16;
424     ACPI_RESOURCE_ADDRESS32     *Address32;


427     if (!Resource || !Out)
428     {
429         return (AE_BAD_PARAMETER);
430     }

432     /* Convert 16 or 32 address descriptor to 64 */

434     switch (Resource->Type)
435     {
436     case ACPI_RESOURCE_TYPE_ADDRESS16:

438         Address16 = ACPI_CAST_PTR (ACPI_RESOURCE_ADDRESS16, &Resource->Data);
439         ACPI_COPY_ADDRESS (Out, Address16);
440         break;

442     case ACPI_RESOURCE_TYPE_ADDRESS32:

444         Address32 = ACPI_CAST_PTR (ACPI_RESOURCE_ADDRESS32, &Resource->Data);
445         ACPI_COPY_ADDRESS (Out, Address32);
446         break;

448     case ACPI_RESOURCE_TYPE_ADDRESS64:

450         /* Simple copy for 64 bit source */

452         ACPI_MEMCPY (Out, &Resource->Data, sizeof (ACPI_RESOURCE_ADDRESS64));
453         break;

455     default:
```

```
457         return (AE_BAD_PARAMETER);
458     }

460     return (AE_OK);
461 }

463 ACPI_EXPORT_SYMBOL (AcpiResourceToAddress64)


466 /*******************************************************************************
467  *
468  * FUNCTION:    AcpiGetVendorResource
469  *
470  * PARAMETERS:  DeviceHandle    - Handle for the parent device object
471  *              Name            - Method name for the parent resource
472  *                                (METHOD_NAME__CRS or METHOD_NAME__PRS)
473  *              Uuid            - Pointer to the UUID to be matched.
474  *                                includes both subtype and 16-byte UUID
475  *              RetBuffer       - Where the vendor resource is returned
476  *
477  * RETURN:      Status
478  *
479  * DESCRIPTION: Walk a resource template for the specified device to find a
431  * DESCRIPTION: Walk a resource template for the specified evice to find a
480  *              vendor-defined resource that matches the supplied UUID and
481  *              UUID subtype. Returns a ACPI_RESOURCE of type Vendor.
482  *
483  ******************************************************************************/

485 ACPI_STATUS
486 AcpiGetVendorResource (
487     ACPI_HANDLE             DeviceHandle,
488     char                    *Name,
489     ACPI_VENDOR_UUID        *Uuid,
490     ACPI_BUFFER             *RetBuffer)
491 {
492     ACPI_VENDOR_WALK_INFO   Info;
493     ACPI_STATUS             Status;


496     /* Other parameters are validated by AcpiWalkResources */

498     if (!Uuid || !RetBuffer)
499     {
500         return (AE_BAD_PARAMETER);
501     }

503     Info.Uuid = Uuid;
504     Info.Buffer = RetBuffer;
505     Info.Status = AE_NOT_EXIST;

507     /* Walk the _CRS or _PRS resource list for this device */

509     Status = AcpiWalkResources (DeviceHandle, Name, AcpiRsMatchVendorResource,
510                 &Info);
511     if (ACPI_FAILURE (Status))
512     {
513         return (Status);
514     }

516     return (Info.Status);
517 }
_____unchanged_portion_omitted_


589 /*******************************************************************************
```



```
590  *
591  * FUNCTION:    AcpiWalkResourceBuffer
543  * FUNCTION:    AcpiWalkResources
592  *
593  * PARAMETERS:  Buffer          - Formatted buffer returned by one of the
594  *                                various Get*Resource functions
545  * PARAMETERS:  DeviceHandle    - Handle to the device object for the
546  *                                device we are querying
547  *              Name            - Method name of the resources we want
548  *                                (METHOD_NAME__CRS or METHOD_NAME__PRS)
595  *              UserFunction    - Called for each resource
596  *              Context         - Passed to UserFunction
597  *
598  * RETURN:      Status
599  *
600  * DESCRIPTION: Walks the input resource template. The UserFunction is called
601  *              once for each resource in the list.
554  * DESCRIPTION: Retrieves the current or possible resource list for the
555  *              specified device. The UserFunction is called once for
556  *              each resource in the list.
602  *
603  ******************************************************************************/

605 ACPI_STATUS
606 AcpiWalkResourceBuffer (
607     ACPI_BUFFER                 *Buffer,
561 AcpiWalkResources (
562     ACPI_HANDLE                 DeviceHandle,
563     char                        *Name,
608     ACPI_WALK_RESOURCE_CALLBACK UserFunction,
609     void                        *Context)
610 {
611     ACPI_STATUS                 Status = AE_OK;
567     ACPI_STATUS                 Status;
568     ACPI_BUFFER                 Buffer;
612     ACPI_RESOURCE               *Resource;
613     ACPI_RESOURCE               *ResourceEnd;


616     ACPI_FUNCTION_TRACE (AcpiWalkResourceBuffer);
573     ACPI_FUNCTION_TRACE (AcpiWalkResources);


619     /* Parameter validation */

621     if (!Buffer || !Buffer->Pointer || !UserFunction)
578     if (!DeviceHandle || !UserFunction || !Name ||
579         (!ACPI_COMPARE_NAME (Name, METHOD_NAME__CRS) &&
580          !ACPI_COMPARE_NAME (Name, METHOD_NAME__PRS)))
622     {
623         return_ACPI_STATUS (AE_BAD_PARAMETER);
624     }

626     /* Buffer contains the resource list and length */
585     /* Get the _CRS or _PRS resource list */

628     Resource = ACPI_CAST_PTR (ACPI_RESOURCE, Buffer->Pointer);
629     ResourceEnd = ACPI_ADD_PTR (ACPI_RESOURCE, Buffer->Pointer, Buffer->Length);
587     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
588     Status = AcpiRsGetMethodData (DeviceHandle, Name, &Buffer);
589     if (ACPI_FAILURE (Status))
590     {
591         return_ACPI_STATUS (Status);
592     }

594     /* Buffer now contains the resource list */
```

```
596     Resource = ACPI_CAST_PTR (ACPI_RESOURCE, Buffer.Pointer);
597     ResourceEnd = ACPI_ADD_PTR (ACPI_RESOURCE, Buffer.Pointer, Buffer.Length);

631     /* Walk the resource list until the EndTag is found (or buffer end) */

633     while (Resource < ResourceEnd)
634     {
635         /* Sanity check the resource type */
603         /* Sanity check the resource */

637         if (Resource->Type > ACPI_RESOURCE_TYPE_MAX)
638         {
639             Status = AE_AML_INVALID_RESOURCE_TYPE;
640             break;
641         }

643         /* Sanity check the length. It must not be zero, or we loop forever */

645         if (!Resource->Length)
646         {
647             return_ACPI_STATUS (AE_AML_BAD_RESOURCE_LENGTH);
648         }

650         /* Invoke the user function, abort on any error returned */

652         Status = UserFunction (Resource, Context);
653         if (ACPI_FAILURE (Status))
654         {
655             if (Status == AE_CTRL_TERMINATE)
656             {
657                 /* This is an OK termination by the user function */

659                 Status = AE_OK;
660             }
661             break;
662         }

664         /* EndTag indicates end-of-list */

666         if (Resource->Type == ACPI_RESOURCE_TYPE_END_TAG)
667         {
668             break;
669         }

671         /* Get the next resource descriptor */

673         Resource = ACPI_NEXT_RESOURCE (Resource);
634         Resource = ACPI_ADD_PTR (ACPI_RESOURCE, Resource, Resource->Length);
674     }

676     return_ACPI_STATUS (Status);
677 }

679 ACPI_EXPORT_SYMBOL (AcpiWalkResourceBuffer)


682 /*******************************************************************************
683  *
684  * FUNCTION:    AcpiWalkResources
685  *
686  * PARAMETERS:  DeviceHandle    - Handle to the device object for the
687  *                                device we are querying
688  *              Name            - Method name of the resources we want.
689  *                                (METHOD_NAME__CRS, METHOD_NAME__PRS, or
690  *                                METHOD_NAME__AEI)
```



```
691  *              UserFunction    - Called for each resource
692  *              Context         - Passed to UserFunction
693  *
694  * RETURN:      Status
695  *
696  * DESCRIPTION: Retrieves the current or possible resource list for the
697  *              specified device. The UserFunction is called once for
698  *              each resource in the list.
699  *
700  ******************************************************************************/

702 ACPI_STATUS
703 AcpiWalkResources (
704     ACPI_HANDLE                 DeviceHandle,
705     char                        *Name,
706     ACPI_WALK_RESOURCE_CALLBACK UserFunction,
707     void                        *Context)
708 {
709     ACPI_STATUS                 Status;
710     ACPI_BUFFER                 Buffer;


713     ACPI_FUNCTION_TRACE (AcpiWalkResources);


716     /* Parameter validation */

718     if (!DeviceHandle || !UserFunction || !Name ||
719         (!ACPI_COMPARE_NAME (Name, METHOD_NAME__CRS) &&
720          !ACPI_COMPARE_NAME (Name, METHOD_NAME__PRS) &&
721          !ACPI_COMPARE_NAME (Name, METHOD_NAME__AEI)))
722     {
723         return_ACPI_STATUS (AE_BAD_PARAMETER);
724     }

726     /* Get the _CRS/_PRS/_AEI resource list */

728     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
729     Status = AcpiRsGetMethodData (DeviceHandle, Name, &Buffer);
730     if (ACPI_FAILURE (Status))
731     {
732         return_ACPI_STATUS (Status);
733     }

735     /* Walk the resource list and cleanup */

737     Status = AcpiWalkResourceBuffer (&Buffer, UserFunction, Context);
738     ACPI_FREE (Buffer.Pointer);
739     return_ACPI_STATUS (Status);
740 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   25213 Thu Dec 26 13:49:33 2013
new/usr/src/common/acpica/components/tables/tbfadt.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: tbfadt   - FADT table utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __TBFADT_C__

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#define _COMPONENT          ACPI_TABLES
        ACPI_MODULE_NAME    ("tbfadt")

/* Local prototypes */

static void
AcpiTbInitGenericAddress (
    ACPI_GENERIC_ADDRESS    *GenericAddress,
    UINT8                   SpaceId,
    UINT8                   ByteWidth,
```



```
    UINT64                  Address,
    char                    *RegisterName);

static void
AcpiTbConvertFadt (
    void);

static void
AcpiTbSetupFadtRegisters (
    void);

static UINT64
AcpiTbSelectAddress (
    char                    *RegisterName,
    UINT32                  Address32,
    UINT64                  Address64);


/* Table for conversion of FADT to common internal format and FADT validation */

typedef struct acpi_fadt_info
{
    char                    *Name;
    UINT16                  Address64;
    UINT16                  Address32;
    UINT16                  Length;
    UINT8                   DefaultLength;
    UINT8                   Type;

} ACPI_FADT_INFO;

#define ACPI_FADT_OPTIONAL          0
#define ACPI_FADT_REQUIRED          1
#define ACPI_FADT_SEPARATE_LENGTH   2

static ACPI_FADT_INFO     FadtInfoTable[] =
{
    {"Pm1aEventBlock",
        ACPI_FADT_OFFSET (XPm1aEventBlock),
        ACPI_FADT_OFFSET (Pm1aEventBlock),
        ACPI_FADT_OFFSET (Pm1EventLength),
        ACPI_PM1_REGISTER_WIDTH * 2,        /* Enable + Status register */
        ACPI_FADT_REQUIRED},

    {"Pm1bEventBlock",
        ACPI_FADT_OFFSET (XPm1bEventBlock),
        ACPI_FADT_OFFSET (Pm1bEventBlock),
        ACPI_FADT_OFFSET (Pm1EventLength),
        ACPI_PM1_REGISTER_WIDTH * 2,        /* Enable + Status register */
        ACPI_FADT_OPTIONAL},

    {"Pm1aControlBlock",
        ACPI_FADT_OFFSET (XPm1aControlBlock),
        ACPI_FADT_OFFSET (Pm1aControlBlock),
        ACPI_FADT_OFFSET (Pm1ControlLength),
        ACPI_PM1_REGISTER_WIDTH,
        ACPI_FADT_REQUIRED},

    {"Pm1bControlBlock",
        ACPI_FADT_OFFSET (XPm1bControlBlock),
        ACPI_FADT_OFFSET (Pm1bControlBlock),
        ACPI_FADT_OFFSET (Pm1ControlLength),
        ACPI_PM1_REGISTER_WIDTH,
        ACPI_FADT_OPTIONAL},

    {"Pm2ControlBlock",
```

```
126         ACPI_FADT_OFFSET (XPm2ControlBlock),
127         ACPI_FADT_OFFSET (Pm2ControlBlock),
128         ACPI_FADT_OFFSET (Pm2ControlLength),
129         ACPI_PM2_REGISTER_WIDTH,
130         ACPI_FADT_SEPARATE_LENGTH},

132     {"PmTimerBlock",
133         ACPI_FADT_OFFSET (XPmTimerBlock),
134         ACPI_FADT_OFFSET (PmTimerBlock),
135         ACPI_FADT_OFFSET (PmTimerLength),
136         ACPI_PM_TIMER_WIDTH,
137         ACPI_FADT_SEPARATE_LENGTH},         /* ACPI 5.0A: Timer is optional */

139     {"Gpe0Block",
140         ACPI_FADT_OFFSET (XGpe0Block),
141         ACPI_FADT_OFFSET (Gpe0Block),
142         ACPI_FADT_OFFSET (Gpe0BlockLength),
143         0,
144         ACPI_FADT_SEPARATE_LENGTH},

146     {"Gpe1Block",
147         ACPI_FADT_OFFSET (XGpe1Block),
148         ACPI_FADT_OFFSET (Gpe1Block),
149         ACPI_FADT_OFFSET (Gpe1BlockLength),
150         0,
151         ACPI_FADT_SEPARATE_LENGTH}
152 };

154 #define ACPI_FADT_INFO_ENTRIES \
155             (sizeof (FadtInfoTable) / sizeof (ACPI_FADT_INFO))


158 /* Table used to split Event Blocks into separate status/enable registers */

160 typedef struct acpi_fadt_pm_info
161 {
162     ACPI_GENERIC_ADDRESS    *Target;
163     UINT16                  Source;
164     UINT8                   RegisterNum;

166 } ACPI_FADT_PM_INFO;

168 static ACPI_FADT_PM_INFO    FadtPmInfoTable[] =
169 {
170     {&AcpiGbl_XPm1aStatus,
171         ACPI_FADT_OFFSET (XPm1aEventBlock),
172         0},

174     {&AcpiGbl_XPm1aEnable,
175         ACPI_FADT_OFFSET (XPm1aEventBlock),
176         1},

178     {&AcpiGbl_XPm1bStatus,
179         ACPI_FADT_OFFSET (XPm1bEventBlock),
180         0},

182     {&AcpiGbl_XPm1bEnable,
183         ACPI_FADT_OFFSET (XPm1bEventBlock),
184         1}
185 };

187 #define ACPI_FADT_PM_INFO_ENTRIES \
188             (sizeof (FadtPmInfoTable) / sizeof (ACPI_FADT_PM_INFO))


191 /******************************************************************************
```



```
192  *
193  * FUNCTION:    AcpiTbInitGenericAddress
194  *
195  * PARAMETERS:  GenericAddress      - GAS struct to be initialized
196  *              SpaceId             - ACPI Space ID for this register
197  *              ByteWidth           - Width of this register
198  *              Address             - Address of the register
199  *              RegisterName        - ASCII name of the ACPI register
200  *
201  * RETURN:      None
202  *
203  * DESCRIPTION: Initialize a Generic Address Structure (GAS)
204  *              See the ACPI specification for a full description and
205  *              definition of this structure.
206  *
207  *****************************************************************************/

209 static void
210 AcpiTbInitGenericAddress (
211     ACPI_GENERIC_ADDRESS    *GenericAddress,
212     UINT8                   SpaceId,
213     UINT8                   ByteWidth,
214     UINT64                  Address,
215     char                    *RegisterName)
216 {
217     UINT8                   BitWidth;


220     /* Bit width field in the GAS is only one byte long, 255 max */

222     BitWidth = (UINT8) (ByteWidth * 8);

224     if (ByteWidth > 31) /* (31*8)=248 */
225     {
226         ACPI_ERROR ((AE_INFO,
227             "%s - 32-bit FADT register is too long (%u bytes, %u bits) "
228             "to convert to GAS struct - 255 bits max, truncating",
229             RegisterName, ByteWidth, (ByteWidth * 8)));

231         BitWidth = 255;
232     }

234     /*
235      * The 64-bit Address field is non-aligned in the byte packed
236      * GAS struct.
237      */
238     ACPI_MOVE_64_TO_64 (&GenericAddress->Address, &Address);

240     /* All other fields are byte-wide */

242     GenericAddress->SpaceId = SpaceId;
243     GenericAddress->BitWidth = BitWidth;
244     GenericAddress->BitOffset = 0;
245     GenericAddress->AccessWidth = 0; /* Access width ANY */
246 }


249 /******************************************************************************
250  *
251  * FUNCTION:    AcpiTbSelectAddress
252  *
253  * PARAMETERS:  RegisterName        - ASCII name of the ACPI register
254  *              Address32           - 32-bit address of the register
255  *              Address64           - 64-bit address of the register
256  *
257  * RETURN:      The resolved 64-bit address
```

```
 *
 * DESCRIPTION: Select between 32-bit and 64-bit versions of addresses within
 *              the FADT. Used for the FACS and DSDT addresses.
 *
 * NOTES:
 *
 * Check for FACS and DSDT address mismatches. An address mismatch between
 * the 32-bit and 64-bit address fields (FIRMWARE_CTRL/X_FIRMWARE_CTRL and
 * DSDT/X_DSDT) could be a corrupted address field or it might indicate
 * the presence of two FACS or two DSDT tables.
 *
 * November 2013:
 * By default, as per the ACPICA specification, a valid 64-bit address is
 * used regardless of the value of the 32-bit address. However, this
 * behavior can be overridden via the AcpiGbl_Use32BitFadtAddresses flag.
 *
 *****************************************************************************/

static UINT64
AcpiTbSelectAddress (
    char                    *RegisterName,
    UINT32                  Address32,
    UINT64                  Address64)
{

    if (!Address64)
    {
        /* 64-bit address is zero, use 32-bit address */

        return ((UINT64) Address32);
    }

    if (Address32 &&
       (Address64 != (UINT64) Address32))
    {
        /* Address mismatch between 32-bit and 64-bit versions */

        ACPI_BIOS_WARNING ((AE_INFO,
            "32/64X %s address mismatch in FADT: "
            "0x%8.8X/0x%8.8X%8.8X, using %u-bit address",
            RegisterName, Address32, ACPI_FORMAT_UINT64 (Address64),
            AcpiGbl_Use32BitFadtAddresses ? 32 : 64));

        /* 32-bit address override */

        if (AcpiGbl_Use32BitFadtAddresses)
        {
            return ((UINT64) Address32);
        }
    }

    /* Default is to use the 64-bit address */

    return (Address64);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiTbParseFadt
 *
 * PARAMETERS:  TableIndex          - Index for the FADT
 *
 * RETURN:      None
 *
 * DESCRIPTION: Initialize the FADT, DSDT and FACS tables
```



```
 *              (FADT contains the addresses of the DSDT and FACS)
 *
 *****************************************************************************/

void
AcpiTbParseFadt (
    UINT32                  TableIndex)
{
    UINT32                  Length;
    ACPI_TABLE_HEADER       *Table;


    /*
     * The FADT has multiple versions with different lengths,
     * and it contains pointers to both the DSDT and FACS tables.
     *
     * Get a local copy of the FADT and convert it to a common format
     * Map entire FADT, assumed to be smaller than one page.
     */
    Length = AcpiGbl_RootTableList.Tables[TableIndex].Length;

    Table = AcpiOsMapMemory (
                AcpiGbl_RootTableList.Tables[TableIndex].Address, Length);
    if (!Table)
    {
        return;
    }

    /*
     * Validate the FADT checksum before we copy the table. Ignore
     * checksum error as we want to try to get the DSDT and FACS.
     */
    (void) AcpiTbVerifyChecksum (Table, Length);

    /* Create a local copy of the FADT in common ACPI 2.0+ format */

    AcpiTbCreateLocalFadt (Table, Length);

    /* All done with the real FADT, unmap it */

    AcpiOsUnmapMemory (Table, Length);

    /* Obtain the DSDT and FACS tables via their addresses within the FADT */

    AcpiTbInstallTable ((ACPI_PHYSICAL_ADDRESS) AcpiGbl_FADT.XDsdt,
        ACPI_SIG_DSDT, ACPI_TABLE_INDEX_DSDT);

    /* If Hardware Reduced flag is set, there is no FACS */

    if (!AcpiGbl_ReducedHardware)
    {
        AcpiTbInstallTable ((ACPI_PHYSICAL_ADDRESS) AcpiGbl_FADT.XFacs,
            ACPI_SIG_FACS, ACPI_TABLE_INDEX_FACS);
    }
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiTbCreateLocalFadt
 *
 * PARAMETERS:  Table               - Pointer to BIOS FADT
 *              Length              - Length of the table
 *
 * RETURN:      None
 *
```

```
 * DESCRIPTION: Get a local copy of the FADT and convert it to a common format.
 *              Performs validation on some important FADT fields.
 *
 * NOTE:        We create a local copy of the FADT regardless of the version.
 *
 ******************************************************************************/

void
AcpiTbCreateLocalFadt (
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Length)
{

    /*
     * Check if the FADT is larger than the largest table that we expect
     * (the ACPI 5.0 version). If so, truncate the table, and issue
     * a warning.
     */
    if (Length > sizeof (ACPI_TABLE_FADT))
    {
        ACPI_BIOS_WARNING ((AE_INFO,
            "FADT (revision %u) is longer than ACPI 5.0 version, "
            "truncating length %u to %u",
            Table->Revision, Length, (UINT32) sizeof (ACPI_TABLE_FADT)));
    }

    /* Clear the entire local FADT */

    ACPI_MEMSET (&AcpiGbl_FADT, 0, sizeof (ACPI_TABLE_FADT));

    /* Copy the original FADT, up to sizeof (ACPI_TABLE_FADT) */

    ACPI_MEMCPY (&AcpiGbl_FADT, Table,
        ACPI_MIN (Length, sizeof (ACPI_TABLE_FADT)));

    /* Take a copy of the Hardware Reduced flag */

    AcpiGbl_ReducedHardware = FALSE;
    if (AcpiGbl_FADT.Flags & ACPI_FADT_HW_REDUCED)
    {
        AcpiGbl_ReducedHardware = TRUE;
    }

    /* Convert the local copy of the FADT to the common internal format */

    AcpiTbConvertFadt ();

    /* Initialize the global ACPI register structures */

    AcpiTbSetupFadtRegisters ();
}


/******************************************************************************
 *
 * FUNCTION:    AcpiTbConvertFadt
 *
 * PARAMETERS:  None - AcpiGbl_FADT is used.
 *
 * RETURN:      None
 *
 * DESCRIPTION: Converts all versions of the FADT to a common internal format.
 *              Expand 32-bit addresses to 64-bit as necessary. Also validate
 *              important fields within the FADT.
 *
 * NOTE:        AcpiGbl_FADT must be of size (ACPI_TABLE_FADT), and must
```



```
 *              contain a copy of the actual BIOS-provided FADT.
 *
 * Notes on 64-bit register addresses:
 *
 * After this FADT conversion, later ACPICA code will only use the 64-bit "X"
 * fields of the FADT for all ACPI register addresses.
 *
 * The 64-bit X fields are optional extensions to the original 32-bit FADT
 * V1.0 fields. Even if they are present in the FADT, they are optional and
 * are unused if the BIOS sets them to zero. Therefore, we must copy/expand
 * 32-bit V1.0 fields to the 64-bit X fields if the the 64-bit X field is
 * originally zero.
 *
 * For ACPI 1.0 FADTs (that contain no 64-bit addresses), all 32-bit address
 * fields are expanded to the corresponding 64-bit X fields in the internal
 * common FADT.
 *
 * For ACPI 2.0+ FADTs, all valid (non-zero) 32-bit address fields are expanded
 * to the corresponding 64-bit X fields, if the 64-bit field is originally
 * zero. Adhering to the ACPI specification, we completely ignore the 32-bit
 * field if the 64-bit field is valid, regardless of whether the host OS is
 * 32-bit or 64-bit.
 *
 * Possible additional checks:
 *  (AcpiGbl_FADT.Pm1EventLength >= 4)
 *  (AcpiGbl_FADT.Pm1ControlLength >= 2)
 *  (AcpiGbl_FADT.PmTimerLength >= 4)
 *  Gpe block lengths must be multiple of 2
 *
 ******************************************************************************/

static void
AcpiTbConvertFadt (
    void)
{
    char                    *Name;
    ACPI_GENERIC_ADDRESS    *Address64;
    UINT32                  Address32;
    UINT8                   Length;
    UINT32                  i;


    /*
     * For ACPI 1.0 FADTs (revision 1 or 2), ensure that reserved fields which
     * should be zero are indeed zero. This will workaround BIOSs that
     * inadvertently place values in these fields.
     *
     * The ACPI 1.0 reserved fields that will be zeroed are the bytes located
     * at offset 45, 55, 95, and the word located at offset 109, 110.
     *
     * Note: The FADT revision value is unreliable. Only the length can be
     * trusted.
     */
    if (AcpiGbl_FADT.Header.Length <= ACPI_FADT_V2_SIZE)
    {
        AcpiGbl_FADT.PreferredProfile = 0;
        AcpiGbl_FADT.PstateControl = 0;
        AcpiGbl_FADT.CstControl = 0;
        AcpiGbl_FADT.BootFlags = 0;
    }

    /*
     * Now we can update the local FADT length to the length of the
     * current FADT version as defined by the ACPI specification.
     * Thus, we will have a common FADT internally.
     */
```

```
522     AcpiGbl_FADT.Header.Length = sizeof (ACPI_TABLE_FADT);
523
524     /*
525      * Expand the 32-bit FACS and DSDT addresses to 64-bit as necessary.
526      * Later ACPICA code will always use the X 64-bit field.
527      */
528     AcpiGbl_FADT.XFacs = AcpiTbSelectAddress ("FACS",
529         AcpiGbl_FADT.Facs, AcpiGbl_FADT.XFacs);
530
531     AcpiGbl_FADT.XDsdt = AcpiTbSelectAddress ("DSDT",
532         AcpiGbl_FADT.Dsdt, AcpiGbl_FADT.XDsdt);
533
534     /* If Hardware Reduced flag is set, we are all done */
535
536     if (AcpiGbl_ReducedHardware)
537     {
538         return;
539     }
540
541     /* Examine all of the 64-bit extended address fields (X fields) */
542
543     for (i = 0; i < ACPI_FADT_INFO_ENTRIES; i++)
544     {
545         /*
546          * Get the 32-bit and 64-bit addresses, as well as the register
547          * length and register name.
548          */
549         Address32 = *ACPI_ADD_PTR (UINT32,
550             &AcpiGbl_FADT, FadtInfoTable[i].Address32);
551
552         Address64 = ACPI_ADD_PTR (ACPI_GENERIC_ADDRESS,
553             &AcpiGbl_FADT, FadtInfoTable[i].Address64);
554
555         Length = *ACPI_ADD_PTR (UINT8,
556             &AcpiGbl_FADT, FadtInfoTable[i].Length);
557
558         Name = FadtInfoTable[i].Name;
559
560         /*
561          * Expand the ACPI 1.0 32-bit addresses to the ACPI 2.0 64-bit "X"
562          * generic address structures as necessary. Later code will always use
563          * the 64-bit address structures.
564          *
565          * November 2013:
566          * Now always use the 64-bit address if it is valid (non-zero), in
567          * accordance with the ACPI specification which states that a 64-bit
568          * address supersedes the 32-bit version. This behavior can be
569          * overridden by the AcpiGbl_Use32BitFadtAddresses flag.
570          *
571          * During 64-bit address construction and verification,
572          * these cases are handled:
573          *
574          * Address32 zero, Address64 [don't care]   - Use Address64
575          *
576          * Address32 non-zero, Address64 zero       - Copy/use Address32
577          * Address32 non-zero == Address64 non-zero - Use Address64
578          * Address32 non-zero != Address64 non-zero - Warning, use Address64
579          *
580          * Override: if AcpiGbl_Use32BitFadtAddresses is TRUE, and:
581          * Address32 non-zero != Address64 non-zero - Warning, copy/use Address3
582          *
583          * Note: SpaceId is always I/O for 32-bit legacy address fields
584          */
585         if (Address32)
586         {
587             if (!Address64->Address)
```



```
588             {
589                 /* 64-bit address is zero, use 32-bit address */
590
591                 AcpiTbInitGenericAddress (Address64,
592                     ACPI_ADR_SPACE_SYSTEM_IO,
593                     *ACPI_ADD_PTR (UINT8, &AcpiGbl_FADT,
594                         FadtInfoTable[i].Length),
595                     (UINT64) Address32, Name);
596             }
597             else if (Address64->Address != (UINT64) Address32)
598             {
599                 /* Address mismatch */
600
601                 ACPI_BIOS_WARNING ((AE_INFO,
602                     "32/64X address mismatch in FADT/%s: "
603                     "0x%8.8X/0x%8.8X%8.8X, using %u-bit address",
604                     Name, Address32,
605                     ACPI_FORMAT_UINT64 (Address64->Address),
606                     AcpiGbl_Use32BitFadtAddresses ? 32 : 64));
607
608                 if (AcpiGbl_Use32BitFadtAddresses)
609                 {
610                     /* 32-bit address override */
611
612                     AcpiTbInitGenericAddress (Address64,
613                         ACPI_ADR_SPACE_SYSTEM_IO,
614                         *ACPI_ADD_PTR (UINT8, &AcpiGbl_FADT,
615                             FadtInfoTable[i].Length),
616                         (UINT64) Address32, Name);
617                 }
618             }
619         }
620
621         /*
622          * For each extended field, check for length mismatch between the
623          * legacy length field and the corresponding 64-bit X length field.
624          * Note: If the legacy length field is > 0xFF bits, ignore this
625          * check. (GPE registers can be larger than the 64-bit GAS structure
626          * can accomodate, 0xFF bits).
627          */
628         if (Address64->Address &&
629            (ACPI_MUL_8 (Length) <= ACPI_UINT8_MAX) &&
630            (Address64->BitWidth != ACPI_MUL_8 (Length)))
631         {
632             ACPI_BIOS_WARNING ((AE_INFO,
633                 "32/64X length mismatch in FADT/%s: %u/%u",
634                 Name, ACPI_MUL_8 (Length), Address64->BitWidth));
635         }
636
637         if (FadtInfoTable[i].Type & ACPI_FADT_REQUIRED)
638         {
639             /*
640              * Field is required (PM1aEvent, PM1aControl).
641              * Both the address and length must be non-zero.
642              */
643             if (!Address64->Address || !Length)
644             {
645                 ACPI_BIOS_ERROR ((AE_INFO,
646                     "Required FADT field %s has zero address and/or length: "
647                     "0x%8.8X%8.8X/0x%X",
648                     Name, ACPI_FORMAT_UINT64 (Address64->Address), Length));
649             }
650         }
651         else if (FadtInfoTable[i].Type & ACPI_FADT_SEPARATE_LENGTH)
652         {
653             /*
```

```
654              * Field is optional (PM2Control, GPE0, GPE1) AND has its own
655              * length field. If present, both the address and length must
656              * be valid.
657              */
658             if ((Address64->Address && !Length) ||
659                 (!Address64->Address && Length))
660             {
661                 ACPI_BIOS_WARNING ((AE_INFO,
662                     "Optional FADT field %s has zero address or length: "
663                     "0x%8.8X%8.8X/0x%X",
664                     Name, ACPI_FORMAT_UINT64 (Address64->Address), Length));
665             }
666         }
667     }
668 }


671 /******************************************************************************
672  *
673  * FUNCTION:    AcpiTbSetupFadtRegisters
674  *
675  * PARAMETERS:  None, uses AcpiGbl_FADT.
676  *
677  * RETURN:      None
678  *
679  * DESCRIPTION: Initialize global ACPI PM1 register definitions. Optionally,
680  *              force FADT register definitions to their default lengths.
681  *
682  *****************************************************************************/

684 static void
685 AcpiTbSetupFadtRegisters (
686     void)
687 {
688     ACPI_GENERIC_ADDRESS    *Target64;
689     ACPI_GENERIC_ADDRESS    *Source64;
690     UINT8                   Pm1RegisterByteWidth;
691     UINT32                  i;


694     /*
695      * Optionally check all register lengths against the default values and
696      * update them if they are incorrect.
697      */
698     if (AcpiGbl_UseDefaultRegisterWidths)
699     {
700         for (i = 0; i < ACPI_FADT_INFO_ENTRIES; i++)
701         {
702             Target64 = ACPI_ADD_PTR (ACPI_GENERIC_ADDRESS, &AcpiGbl_FADT,
703                 FadtInfoTable[i].Address64);

705             /*
706              * If a valid register (Address != 0) and the (DefaultLength > 0)
707              * (Not a GPE register), then check the width against the default.
708              */
709             if ((Target64->Address) &&
710                 (FadtInfoTable[i].DefaultLength > 0) &&
711                 (FadtInfoTable[i].DefaultLength != Target64->BitWidth))
712             {
713                 ACPI_BIOS_WARNING ((AE_INFO,
714                     "Invalid length for FADT/%s: %u, using default %u",
715                     FadtInfoTable[i].Name, Target64->BitWidth,
716                     FadtInfoTable[i].DefaultLength));

718                 /* Incorrect size, set width to the default */
```



```
720                 Target64->BitWidth = FadtInfoTable[i].DefaultLength;
721             }
722         }
723     }

725     /*
726      * Get the length of the individual PM1 registers (enable and status).
727      * Each register is defined to be (event block length / 2). Extra divide
728      * by 8 converts bits to bytes.
729      */
730     Pm1RegisterByteWidth = (UINT8)
731         ACPI_DIV_16 (AcpiGbl_FADT.XPm1aEventBlock.BitWidth);

733     /*
734      * Calculate separate GAS structs for the PM1x (A/B) Status and Enable
735      * registers. These addresses do not appear (directly) in the FADT, so it
736      * is useful to pre-calculate them from the PM1 Event Block definitions.
737      *
738      * The PM event blocks are split into two register blocks, first is the
739      * PM Status Register block, followed immediately by the PM Enable
740      * Register block. Each is of length (Pm1EventLength/2)
741      *
742      * Note: The PM1A event block is required by the ACPI specification.
743      * However, the PM1B event block is optional and is rarely, if ever,
744      * used.
745      */

747     for (i = 0; i < ACPI_FADT_PM_INFO_ENTRIES; i++)
748     {
749         Source64 = ACPI_ADD_PTR (ACPI_GENERIC_ADDRESS, &AcpiGbl_FADT,
750             FadtPmInfoTable[i].Source);

752         if (Source64->Address)
753         {
754             AcpiTbInitGenericAddress (FadtPmInfoTable[i].Target,
755                 Source64->SpaceId, Pm1RegisterByteWidth,
756                 Source64->Address +
757                     (FadtPmInfoTable[i].RegisterNum * Pm1RegisterByteWidth),
758                 "PmRegisters");
759         }
760     }
761 }
```

```
**********************************************************
    5211 Thu Dec 26 13:49:33 2013
new/usr/src/common/acpica/components/tables/tbfind.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: tbfind   - find table
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __TBFIND_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "actables.h"

  50 #define _COMPONENT          ACPI_TABLES
  51         ACPI_MODULE_NAME    ("tbfind")


  54 /*******************************************************************************
  55  *
  56  * FUNCTION:    AcpiTbFindTable
  57  *
  58  * PARAMETERS:  Signature           - String with ACPI table signature
  59  *              OemId               - String with the table OEM ID
```



```
  60  *              OemTableId          - String with the OEM Table ID
  61  *              TableIndex          - Where the table index is returned
  62  *
  63  * RETURN:      Status and table index
  64  *
  65  * DESCRIPTION: Find an ACPI table (in the RSDT/XSDT) that matches the
  66  *              Signature, OEM ID and OEM Table ID. Returns an index that can
  67  *              be used to get the table header or entire table.
  68  *
  69  ******************************************************************************/

  71 ACPI_STATUS
  72 AcpiTbFindTable (
  73     char                    *Signature,
  74     char                    *OemId,
  75     char                    *OemTableId,
  76     UINT32                  *TableIndex)
  77 {
  78     UINT32                  i;
  79     ACPI_STATUS             Status;
  80     ACPI_TABLE_HEADER       Header;


  83     ACPI_FUNCTION_TRACE (TbFindTable);


  86     /* Normalize the input strings */

  88     ACPI_MEMSET (&Header, 0, sizeof (ACPI_TABLE_HEADER));
  89     ACPI_MOVE_NAME (Header.Signature, Signature);
  89     ACPI_STRNCPY (Header.Signature, Signature, ACPI_NAME_SIZE);
  90     ACPI_STRNCPY (Header.OemId, OemId, ACPI_OEM_ID_SIZE);
  91     ACPI_STRNCPY (Header.OemTableId, OemTableId, ACPI_OEM_TABLE_ID_SIZE);

  93     /* Search for the table */

  95     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; ++i)
  96     {
  97         if (ACPI_MEMCMP (&(AcpiGbl_RootTableList.Tables[i].Signature),
  98                          Header.Signature, ACPI_NAME_SIZE))
  99         {
 100             /* Not the requested table */

 102             continue;
 103         }

 105         /* Table with matching signature has been found */

 107         if (!AcpiGbl_RootTableList.Tables[i].Pointer)
 108         {
 109             /* Table is not currently mapped, map it */

 111             Status = AcpiTbVerifyTable (&AcpiGbl_RootTableList.Tables[i]);
 112             if (ACPI_FAILURE (Status))
 113             {
 114                 return_ACPI_STATUS (Status);
 115             }

 117             if (!AcpiGbl_RootTableList.Tables[i].Pointer)
 118             {
 119                 continue;
 120             }
 121         }

 123         /* Check for table match on all IDs */
```

```
125         if (!ACPI_MEMCMP (AcpiGbl_RootTableList.Tables[i].Pointer->Signature,
126                           Header.Signature, ACPI_NAME_SIZE) &&
127             (!OemId[0] ||
128              !ACPI_MEMCMP (AcpiGbl_RootTableList.Tables[i].Pointer->OemId,
129                            Header.OemId, ACPI_OEM_ID_SIZE)) &&
130             (!OemTableId[0] ||
131              !ACPI_MEMCMP (AcpiGbl_RootTableList.Tables[i].Pointer->OemTableId,
132                            Header.OemTableId, ACPI_OEM_TABLE_ID_SIZE)))
133         {
134             *TableIndex = i;

136             ACPI_DEBUG_PRINT ((ACPI_DB_TABLES, "Found table [%4.4s]\n",
137                 Header.Signature));
138             return_ACPI_STATUS (AE_OK);
139         }
140     }

142     return_ACPI_STATUS (AE_NOT_FOUND);
143 }
```

unchanged_portion_omitted_

```
**********************************************************
   23377 Thu Dec 26 13:49:33 2013
new/usr/src/common/acpica/components/tables/tbinstal.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: tbinstal - ACPI table installation and removal
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __TBINSTAL_C__

  47 #include "acpi.h"
  48 #include "accommon.h"
  49 #include "acnamesp.h"
  50 #include "actables.h"


  53 #define _COMPONENT          ACPI_TABLES
  54         ACPI_MODULE_NAME    ("tbinstal")


  57 /******************************************************************************
  58  *
  59  * FUNCTION:    AcpiTbVerifyTable
```



```
  60  *
  61  * PARAMETERS:  TableDesc           - table
  62  *
  63  * RETURN:      Status
  64  *
  65  * DESCRIPTION: this function is called to verify and map table
  66  *
  67  *****************************************************************************/

  69 ACPI_STATUS
  70 AcpiTbVerifyTable (
  71     ACPI_TABLE_DESC         *TableDesc)
  72 {
  73     ACPI_STATUS             Status = AE_OK;


  76     ACPI_FUNCTION_TRACE (TbVerifyTable);


  79     /* Map the table if necessary */

  81     if (!TableDesc->Pointer)
  82     {
  83         if ((TableDesc->Flags & ACPI_TABLE_ORIGIN_MASK) ==
  84             ACPI_TABLE_ORIGIN_MAPPED)
  85         {
  86             TableDesc->Pointer = AcpiOsMapMemory (
  87                 TableDesc->Address, TableDesc->Length);
  88         }

  90         if (!TableDesc->Pointer)
  91         {
  92             return_ACPI_STATUS (AE_NO_MEMORY);
  93         }
  94     }

  96     /* FACS is the odd table, has no standard ACPI header and no checksum */

  98     if (!ACPI_COMPARE_NAME (&TableDesc->Signature, ACPI_SIG_FACS))
  99     {
  96     /* Always calculate checksum, ignore bad checksum if requested */

  98     Status = AcpiTbVerifyChecksum (TableDesc->Pointer, TableDesc->Length);
 103     }

 100     return_ACPI_STATUS (Status);
 101 }


 104 /******************************************************************************
 105  *
 106  * FUNCTION:    AcpiTbAddTable
 107  *
 108  * PARAMETERS:  TableDesc           - Table descriptor
 109  *              TableIndex          - Where the table index is returned
 110  *
 111  * RETURN:      Status
 112  *
 113  * DESCRIPTION: This function is called to add an ACPI table. It is used to
 114  *              dynamically load tables via the Load and LoadTable AML
 115  *              operators.
 116  *
 117  *****************************************************************************/

 119 ACPI_STATUS
 120 AcpiTbAddTable (
```

```
121     ACPI_TABLE_DESC         *TableDesc,
122     UINT32                  *TableIndex)
123 {
124     UINT32                  i;
125     ACPI_STATUS             Status = AE_OK;
131     ACPI_TABLE_HEADER       *OverrideTable = NULL;


128     ACPI_FUNCTION_TRACE (TbAddTable);


131     if (!TableDesc->Pointer)
132     {
133         Status = AcpiTbVerifyTable (TableDesc);
134         if (ACPI_FAILURE (Status) || !TableDesc->Pointer)
135         {
136             return_ACPI_STATUS (Status);
137         }
138     }

140     /*
141      * Validate the incoming table signature.
142      *
143      * 1) Originally, we checked the table signature for "SSDT" or "PSDT".
144      * 2) We added support for OEMx tables, signature "OEM".
145      * 3) Valid tables were encountered with a null signature, so we just
146      *    gave up on validating the signature, (05/2008).
147      * 4) We encountered non-AML tables such as the MADT, which caused
148      *    interpreter errors and kernel faults. So now, we once again allow
149      *    only "SSDT", "OEMx", and now, also a null signature. (05/2011).
150      */
151     if ((TableDesc->Pointer->Signature[0] != 0x00) &&
152        (!ACPI_COMPARE_NAME (TableDesc->Pointer->Signature, ACPI_SIG_SSDT)) &&
153        (ACPI_STRNCMP (TableDesc->Pointer->Signature, "OEM", 3)))
154     {
155         ACPI_BIOS_ERROR ((AE_INFO,
156             "Table has invalid signature [%4.4s] (0x%8.8X), "
157             "must be SSDT or OEMx",
158             AcpiUtValidAcpiName (TableDesc->Pointer->Signature) ?
161         ACPI_ERROR ((AE_INFO,
162             "Table has invalid signature [%4.4s] (0x%8.8X), must be SSDT or OEMx
163             AcpiUtValidAcpiName (*(UINT32 *) TableDesc->Pointer->Signature) ?
159                 TableDesc->Pointer->Signature : "????",
160             *(UINT32 *) TableDesc->Pointer->Signature));

162         return_ACPI_STATUS (AE_BAD_SIGNATURE);
163     }

165     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);

167     /* Check if table is already registered */

169     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; ++i)
170     {
171         if (!AcpiGbl_RootTableList.Tables[i].Pointer)
172         {
173             Status = AcpiTbVerifyTable (&AcpiGbl_RootTableList.Tables[i]);
174             if (ACPI_FAILURE (Status) ||
175                 !AcpiGbl_RootTableList.Tables[i].Pointer)
176             {
177                 continue;
178             }
179         }

181         /*
182          * Check for a table match on the entire table length,
```



```
183          * not just the header.
184          */
185         if (TableDesc->Length != AcpiGbl_RootTableList.Tables[i].Length)
186         {
187             continue;
188         }

190         if (ACPI_MEMCMP (TableDesc->Pointer,
191                 AcpiGbl_RootTableList.Tables[i].Pointer,
192                 AcpiGbl_RootTableList.Tables[i].Length))
193         {
194             continue;
195         }

197         /*
198          * Note: the current mechanism does not unregister a table if it is
199          * dynamically unloaded. The related namespace entries are deleted,
200          * but the table remains in the root table list.
201          *
202          * The assumption here is that the number of different tables that
203          * will be loaded is actually small, and there is minimal overhead
204          * in just keeping the table in case it is needed again.
205          *
206          * If this assumption changes in the future (perhaps on large
207          * machines with many table load/unload operations), tables will
208          * need to be unregistered when they are unloaded, and slots in the
209          * root table list should be reused when empty.
210          */

212         /*
213          * Table is already registered.
214          * We can delete the table that was passed as a parameter.
215          */
216         AcpiTbDeleteTable (TableDesc);
217         *TableIndex = i;

219         if (AcpiGbl_RootTableList.Tables[i].Flags & ACPI_TABLE_IS_LOADED)
220         {
221             /* Table is still loaded, this is an error */

223             Status = AE_ALREADY_EXISTS;
224             goto Release;
225         }
226         else
227         {
228             /* Table was unloaded, allow it to be reloaded */

230             TableDesc->Pointer = AcpiGbl_RootTableList.Tables[i].Pointer;
231             TableDesc->Address = AcpiGbl_RootTableList.Tables[i].Address;
232             Status = AE_OK;
233             goto PrintHeader;
234         }
235     }

237     /*
238      * ACPI Table Override:
239      * Allow the host to override dynamically loaded tables.
240      * NOTE: the table is fully mapped at this point, and the mapping will
241      * be deleted by TbTableOverride if the table is actually overridden.
242      */
243     (void) AcpiTbTableOverride (TableDesc->Pointer, TableDesc);
246     Status = AcpiOsTableOverride (TableDesc->Pointer, &OverrideTable);
247     if (ACPI_SUCCESS (Status) && OverrideTable)
248     {
249         ACPI_INFO ((AE_INFO,
250             "%4.4s @ 0x%p Table override, replaced with:",
```

```
            TableDesc->Pointer->Signature,
            ACPI_CAST_PTR (void, TableDesc->Address)));

        /* We can delete the table that was passed as a parameter */

        AcpiTbDeleteTable (TableDesc);

        /* Setup descriptor for the new table */

        TableDesc->Address = ACPI_PTR_TO_PHYSADDR (OverrideTable);
        TableDesc->Pointer = OverrideTable;
        TableDesc->Length = OverrideTable->Length;
        TableDesc->Flags = ACPI_TABLE_ORIGIN_OVERRIDE;
    }

    /* Add the table to the global root table list */

    Status = AcpiTbStoreTable (TableDesc->Address, TableDesc->Pointer,
                TableDesc->Length, TableDesc->Flags, TableIndex);
    if (ACPI_FAILURE (Status))
    {
        goto Release;
    }

PrintHeader:
    AcpiTbPrintTableHeader (TableDesc->Address, TableDesc->Pointer);

Release:
    (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
    return_ACPI_STATUS (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiTbTableOverride
 *
 * PARAMETERS:  TableHeader         - Header for the original table
 *              TableDesc           - Table descriptor initialized for the
 *                                    original table. May or may not be mapped.
 *
 * RETURN:      Pointer to the entire new table. NULL if table not overridden.
 *              If overridden, installs the new table within the input table
 *              descriptor.
 *
 * DESCRIPTION: Attempt table override by calling the OSL override functions.
 *              Note: If the table is overridden, then the entire new table
 *              is mapped and returned by this function.
 *
 *****************************************************************************/

ACPI_TABLE_HEADER *
AcpiTbTableOverride (
    ACPI_TABLE_HEADER       *TableHeader,
    ACPI_TABLE_DESC         *TableDesc)
{
    ACPI_STATUS             Status;
    ACPI_TABLE_HEADER       *NewTable = NULL;
    ACPI_PHYSICAL_ADDRESS   NewAddress = 0;
    UINT32                  NewTableLength = 0;
    UINT8                   NewFlags;
    char                    *OverrideType;


    /* (1) Attempt logical override (returns a logical address) */
```



```
    Status = AcpiOsTableOverride (TableHeader, &NewTable);
    if (ACPI_SUCCESS (Status) && NewTable)
    {
        NewAddress = ACPI_PTR_TO_PHYSADDR (NewTable);
        NewTableLength = NewTable->Length;
        NewFlags = ACPI_TABLE_ORIGIN_OVERRIDE;
        OverrideType = "Logical";
        goto FinishOverride;
    }

    /* (2) Attempt physical override (returns a physical address) */

    Status = AcpiOsPhysicalTableOverride (TableHeader,
        &NewAddress, &NewTableLength);
    if (ACPI_SUCCESS (Status) && NewAddress && NewTableLength)
    {
        /* Map the entire new table */

        NewTable = AcpiOsMapMemory (NewAddress, NewTableLength);
        if (!NewTable)
        {
            ACPI_EXCEPTION ((AE_INFO, AE_NO_MEMORY,
                "%4.4s %p Attempted physical table override failed",
                TableHeader->Signature,
                ACPI_CAST_PTR (void, TableDesc->Address)));
            return (NULL);
        }

        OverrideType = "Physical";
        NewFlags = ACPI_TABLE_ORIGIN_MAPPED;
        goto FinishOverride;
    }

    return (NULL); /* There was no override */


FinishOverride:

    ACPI_INFO ((AE_INFO,
        "%4.4s %p %s table override, new table: %p",
        TableHeader->Signature,
        ACPI_CAST_PTR (void, TableDesc->Address),
        OverrideType, NewTable));

    /* We can now unmap/delete the original table (if fully mapped) */

    AcpiTbDeleteTable (TableDesc);

    /* Setup descriptor for the new table */

    TableDesc->Address = NewAddress;
    TableDesc->Pointer = NewTable;
    TableDesc->Length = NewTableLength;
    TableDesc->Flags = NewFlags;

    return (NewTable);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiTbResizeRootTableList
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
```

```
 362  *
 363  * DESCRIPTION: Expand the size of global table array
 364  *
 365  ******************************************************************************/

 367 ACPI_STATUS
 368 AcpiTbResizeRootTableList (
 369     void)
 370 {
 371     ACPI_TABLE_DESC         *Tables;
 372     UINT32                  TableCount;


 375     ACPI_FUNCTION_TRACE (TbResizeRootTableList);


 378     /* AllowResize flag is a parameter to AcpiInitializeTables */

 380     if (!(AcpiGbl_RootTableList.Flags & ACPI_ROOT_ALLOW_RESIZE))
 381     {
 382         ACPI_ERROR ((AE_INFO, "Resize of Root Table Array is not allowed"));
 383         return_ACPI_STATUS (AE_SUPPORT);
 384     }

 386     /* Increase the Table Array size */

 388     if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
 389     {
 390         TableCount = AcpiGbl_RootTableList.MaxTableCount;
 391     }
 392     else
 393     {
 394         TableCount = AcpiGbl_RootTableList.CurrentTableCount;
 395     }

 397     Tables = ACPI_ALLOCATE_ZEROED (
 398         ((ACPI_SIZE) TableCount + ACPI_ROOT_TABLE_SIZE_INCREMENT) *
 317         ((ACPI_SIZE) AcpiGbl_RootTableList.MaxTableCount +
 318             ACPI_ROOT_TABLE_SIZE_INCREMENT) *
 399         sizeof (ACPI_TABLE_DESC));
 400     if (!Tables)
 401     {
 402         ACPI_ERROR ((AE_INFO, "Could not allocate new root table array"));
 403         return_ACPI_STATUS (AE_NO_MEMORY);
 404     }

 406     /* Copy and free the previous table array */

 408     if (AcpiGbl_RootTableList.Tables)
 409     {
 410         ACPI_MEMCPY (Tables, AcpiGbl_RootTableList.Tables,
 411             (ACPI_SIZE) TableCount * sizeof (ACPI_TABLE_DESC));
 331             (ACPI_SIZE) AcpiGbl_RootTableList.MaxTableCount * sizeof (ACPI_TABLE

 413         if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
 414         {
 415             ACPI_FREE (AcpiGbl_RootTableList.Tables);
 416         }
 417     }

 419     AcpiGbl_RootTableList.Tables = Tables;
 420     AcpiGbl_RootTableList.MaxTableCount =
 421         TableCount + ACPI_ROOT_TABLE_SIZE_INCREMENT;
 422     AcpiGbl_RootTableList.Flags |= ACPI_ROOT_ORIGIN_ALLOCATED;
 340     AcpiGbl_RootTableList.MaxTableCount += ACPI_ROOT_TABLE_SIZE_INCREMENT;
 341     AcpiGbl_RootTableList.Flags |= (UINT8) ACPI_ROOT_ORIGIN_ALLOCATED;
```



```
 424     return_ACPI_STATUS (AE_OK);
 425 }
______unchanged_portion_omitted_


 485 /******************************************************************************
 486  *
 487  * FUNCTION:    AcpiTbDeleteTable
 488  *
 489  * PARAMETERS:  TableIndex          - Table index
 490  *
 491  * RETURN:      None
 492  *
 493  * DESCRIPTION: Delete one internal ACPI table
 494  *
 495  *****************************************************************************/

 497 void
 498 AcpiTbDeleteTable (
 499     ACPI_TABLE_DESC         *TableDesc)
 500 {

 502     /* Table must be mapped or allocated */

 504     if (!TableDesc->Pointer)
 505     {
 506         return;
 507     }

 509     switch (TableDesc->Flags & ACPI_TABLE_ORIGIN_MASK)
 510     {
 511     case ACPI_TABLE_ORIGIN_MAPPED:

 513         AcpiOsUnmapMemory (TableDesc->Pointer, TableDesc->Length);
 514         break;

 516     case ACPI_TABLE_ORIGIN_ALLOCATED:

 518         ACPI_FREE (TableDesc->Pointer);
 519         break;

 521     /* Not mapped or allocated, there is nothing we can do */

 523     default:

 525         return;
 439         break;
 526     }

 528     TableDesc->Pointer = NULL;
 529 }


 532 /******************************************************************************
 533  *
 534  * FUNCTION:    AcpiTbTerminate
 535  *
 536  * PARAMETERS:  None
 537  *
 538  * RETURN:      None
 539  *
 540  * DESCRIPTION: Delete all internal ACPI tables
 541  *
 542  *****************************************************************************/
```

```
544 void
545 AcpiTbTerminate (
546     void)
547 {
548     UINT32                  i;


551     ACPI_FUNCTION_TRACE (TbTerminate);


554     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);

556     /* Delete the individual tables */

558     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
559     {
560         AcpiTbDeleteTable (&AcpiGbl_RootTableList.Tables[i]);
561     }

563     /*
564      * Delete the root table array if allocated locally. Array cannot be
565      * mapped, so we don't need to check for that flag.
566      */
567     if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
568     {
569         ACPI_FREE (AcpiGbl_RootTableList.Tables);
570     }

572     AcpiGbl_RootTableList.Tables = NULL;
573     AcpiGbl_RootTableList.Flags = 0;
574     AcpiGbl_RootTableList.CurrentTableCount = 0;

576     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "ACPI Tables freed\n"));
577     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);

579     return_VOID;
580 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    8532 Thu Dec 26 13:49:34 2013
new/usr/src/common/acpica/components/tables/tbprint.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: tbprint - Table output utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __TBPRINT_C__

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#define _COMPONENT          ACPI_TABLES
        ACPI_MODULE_NAME    ("tbprint")


/* Local prototypes */

static void
AcpiTbFixString (
    char                    *String,
    ACPI_SIZE               Length);
```



```
static void
AcpiTbCleanupTableHeader (
    ACPI_TABLE_HEADER       *OutHeader,
    ACPI_TABLE_HEADER       *Header);


/******************************************************************************
 *
 * FUNCTION:    AcpiTbFixString
 *
 * PARAMETERS:  String              - String to be repaired
 *              Length              - Maximum length
 *
 * RETURN:      None
 *
 * DESCRIPTION: Replace every non-printable or non-ascii byte in the string
 *              with a question mark '?'.
 *
 *****************************************************************************/

static void
AcpiTbFixString (
    char                    *String,
    ACPI_SIZE               Length)
{

    while (Length && *String)
    {
        if (!ACPI_IS_PRINT (*String))
        {
            *String = '?';
        }
        String++;
        Length--;
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiTbCleanupTableHeader
 *
 * PARAMETERS:  OutHeader           - Where the cleaned header is returned
 *              Header              - Input ACPI table header
 *
 * RETURN:      Returns the cleaned header in OutHeader
 *
 * DESCRIPTION: Copy the table header and ensure that all "string" fields in
 *              the header consist of printable characters.
 *
 *****************************************************************************/

static void
AcpiTbCleanupTableHeader (
    ACPI_TABLE_HEADER       *OutHeader,
    ACPI_TABLE_HEADER       *Header)
{

    ACPI_MEMCPY (OutHeader, Header, sizeof (ACPI_TABLE_HEADER));

    AcpiTbFixString (OutHeader->Signature, ACPI_NAME_SIZE);
    AcpiTbFixString (OutHeader->OemId, ACPI_OEM_ID_SIZE);
    AcpiTbFixString (OutHeader->OemTableId, ACPI_OEM_TABLE_ID_SIZE);
    AcpiTbFixString (OutHeader->AslCompilerId, ACPI_NAME_SIZE);
}
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiTbPrintTableHeader
 *
 * PARAMETERS:  Address             - Table physical address
 *              Header              - Table header
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print an ACPI table header. Special cases for FACS and RSDP.
 *
 ******************************************************************************/

void
AcpiTbPrintTableHeader (
    ACPI_PHYSICAL_ADDRESS   Address,
    ACPI_TABLE_HEADER       *Header)
{
    ACPI_TABLE_HEADER       LocalHeader;


    /*
     * The reason that the Address is cast to a void pointer is so that we
     * can use %p which will work properly on both 32-bit and 64-bit hosts.
     */
    if (ACPI_COMPARE_NAME (Header->Signature, ACPI_SIG_FACS))
    {
        /* FACS only has signature and length fields */

        ACPI_INFO ((AE_INFO, "%4.4s %p %06X",
            Header->Signature, ACPI_CAST_PTR (void, Address),
            Header->Length));
    }
    else if (ACPI_VALIDATE_RSDP_SIG (Header->Signature))
    {
        /* RSDP has no common fields */

        ACPI_MEMCPY (LocalHeader.OemId,
            ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->OemId, ACPI_OEM_ID_SIZE);
        AcpiTbFixString (LocalHeader.OemId, ACPI_OEM_ID_SIZE);

        ACPI_INFO ((AE_INFO, "RSDP %p %06X (v%.2d %6.6s)",
            ACPI_CAST_PTR (void, Address),
            (ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Revision > 0) ?
                ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Length : 20,
            ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Revision,
            LocalHeader.OemId));
    }
    else
    {
        /* Standard ACPI table with full common header */

        AcpiTbCleanupTableHeader (&LocalHeader, Header);

        ACPI_INFO ((AE_INFO,
            "%4.4s %p %06X (v%.2d %6.6s %8.8s %08X %4.4s %08X)",
            LocalHeader.Signature, ACPI_CAST_PTR (void, Address),
            LocalHeader.Length, LocalHeader.Revision, LocalHeader.OemId,
            LocalHeader.OemTableId, LocalHeader.OemRevision,
            LocalHeader.AslCompilerId, LocalHeader.AslCompilerRevision));
    }
}


/*******************************************************************************
```



```
 *
 * FUNCTION:    AcpiTbValidateChecksum
 *
 * PARAMETERS:  Table               - ACPI table to verify
 *              Length              - Length of entire table
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Verifies that the table checksums to zero. Optionally returns
 *              exception on bad checksum.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiTbVerifyChecksum (
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Length)
{
    UINT8                   Checksum;


    /*
     * FACS/S3PT:
     * They are the odd tables, have no standard ACPI header and no checksum
     */

    if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_S3PT) ||
        ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_FACS))
    {
        return (AE_OK);
    }

    /* Compute the checksum on the table */

    Checksum = AcpiTbChecksum (ACPI_CAST_PTR (UINT8, Table), Length);

    /* Checksum ok? (should be zero) */

    if (Checksum)
    {
        ACPI_BIOS_WARNING ((AE_INFO,
            "Incorrect checksum in table [%4.4s] - 0x%2.2X, "
            "should be 0x%2.2X",
            Table->Signature, Table->Checksum,
            (UINT8) (Table->Checksum - Checksum)));

#if (ACPI_CHECKSUM_ABORT)
        return (AE_BAD_CHECKSUM);
#endif
    }

    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiTbChecksum
 *
 * PARAMETERS:  Buffer          - Pointer to memory region to be checked
 *              Length          - Length of this memory region
 *
 * RETURN:      Checksum (UINT8)
 *
 * DESCRIPTION: Calculates circular checksum of memory region.
 *
```

```
 *****************************************************************************/

UINT8
AcpiTbChecksum (
    UINT8                   *Buffer,
    UINT32                  Length)
{
    UINT8                   Sum = 0;
    UINT8                   *End = Buffer + Length;


    while (Buffer < End)
    {
        Sum = (UINT8) (Sum + *(Buffer++));
    }

    return (Sum);
}
```

```
**********************************************************
   20805 Thu Dec 26 13:49:34 2013
new/usr/src/common/acpica/components/tables/tbutils.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: tbutils - ACPI Table utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __TBUTILS_C__

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#define _COMPONENT          ACPI_TABLES
        ACPI_MODULE_NAME    ("tbutils")


/* Local prototypes */

static ACPI_STATUS
AcpiTbValidateXsdt (
    ACPI_PHYSICAL_ADDRESS   Address);
```



```
static ACPI_PHYSICAL_ADDRESS
AcpiTbGetRootTableEntry (
    UINT8                   *TableEntry,
    UINT32                  TableEntrySize);


#if (!ACPI_REDUCED_HARDWARE)
/******************************************************************************
 *
 * FUNCTION:    AcpiTbInitializeFacs
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create a permanent mapping for the FADT and save it in a global
 *              for accessing the Global Lock and Firmware Waking Vector
 *
 *****************************************************************************/

ACPI_STATUS
AcpiTbInitializeFacs (
    void)
{
    ACPI_STATUS             Status;


    /* If Hardware Reduced flag is set, there is no FACS */

    if (AcpiGbl_ReducedHardware)
    {
        AcpiGbl_FACS = NULL;
        return (AE_OK);
    }

    Status = AcpiGetTableByIndex (ACPI_TABLE_INDEX_FACS,
                ACPI_CAST_INDIRECT_PTR (ACPI_TABLE_HEADER, &AcpiGbl_FACS));
    return (Status);
}
#endif /* !ACPI_REDUCED_HARDWARE */


/******************************************************************************
 *
 * FUNCTION:    AcpiTbTablesLoaded
 *
 * PARAMETERS:  None
 *
 * RETURN:      TRUE if required ACPI tables are loaded
 *
 * DESCRIPTION: Determine if the minimum required ACPI tables are present
 *              (FADT, FACS, DSDT)
 *
 *****************************************************************************/

BOOLEAN
AcpiTbTablesLoaded (
    void)
{

    if (AcpiGbl_RootTableList.CurrentTableCount >= 3)
    {
        return (TRUE);
    }

    return (FALSE);
```

```
126 }


129 /*******************************************************************************
130  *
131  * FUNCTION:    AcpiTbCheckDsdtHeader
132  *
133  * PARAMETERS:  None
134  *
135  * RETURN:      None
136  *
137  * DESCRIPTION: Quick compare to check validity of the DSDT. This will detect
138  *              if the DSDT has been replaced from outside the OS and/or if
139  *              the DSDT header has been corrupted.
140  *
141  ******************************************************************************/

143 void
144 AcpiTbCheckDsdtHeader (
145     void)
146 {

148     /* Compare original length and checksum to current values */

150     if (AcpiGbl_OriginalDsdtHeader.Length != AcpiGbl_DSDT->Length ||
151         AcpiGbl_OriginalDsdtHeader.Checksum != AcpiGbl_DSDT->Checksum)
152     {
153         ACPI_BIOS_ERROR ((AE_INFO,
154             "The DSDT has been corrupted or replaced - "
155             "old, new headers below"));
156         AcpiTbPrintTableHeader (0, &AcpiGbl_OriginalDsdtHeader);
157         AcpiTbPrintTableHeader (0, AcpiGbl_DSDT);

159         /* Disable further error messages */

161         AcpiGbl_OriginalDsdtHeader.Length = AcpiGbl_DSDT->Length;
162         AcpiGbl_OriginalDsdtHeader.Checksum = AcpiGbl_DSDT->Checksum;
163     }
164 }


167 /*******************************************************************************
168  *
169  * FUNCTION:    AcpiTbCopyDsdt
170  *
171  * PARAMETERS:  TableDesc           - Installed table to copy
172  *
173  * RETURN:      None
174  *
175  * DESCRIPTION: Implements a subsystem option to copy the DSDT to local memory.
176  *              Some very bad BIOSs are known to either corrupt the DSDT or
177  *              install a new, bad DSDT. This copy works around the problem.
178  *
179  ******************************************************************************/

181 ACPI_TABLE_HEADER *
182 AcpiTbCopyDsdt (
183     UINT32                  TableIndex)
184 {
185     ACPI_TABLE_HEADER       *NewTable;
186     ACPI_TABLE_DESC         *TableDesc;


189     TableDesc = &AcpiGbl_RootTableList.Tables[TableIndex];

191     NewTable = ACPI_ALLOCATE (TableDesc->Length);
```



```
192     if (!NewTable)
193     {
194         ACPI_ERROR ((AE_INFO, "Could not copy DSDT of length 0x%X",
195             TableDesc->Length));
196         return (NULL);
197     }

199     ACPI_MEMCPY (NewTable, TableDesc->Pointer, TableDesc->Length);
200     AcpiTbDeleteTable (TableDesc);
201     TableDesc->Pointer = NewTable;
202     TableDesc->Flags = ACPI_TABLE_ORIGIN_ALLOCATED;

204     ACPI_INFO ((AE_INFO,
205         "Forced DSDT copy: length 0x%05X copied locally, original unmapped",
206         NewTable->Length));

208     return (NewTable);
209 }


212 /*******************************************************************************
213  *
214  * FUNCTION:    AcpiTbInstallTable
215  *
216  * PARAMETERS:  Address                 - Physical address of DSDT or FACS
217  *              Signature               - Table signature, NULL if no need to
218  *                                        match
219  *              TableIndex              - Index into root table array
220  *
221  * RETURN:      None
222  *
223  * DESCRIPTION: Install an ACPI table into the global data structure. The
224  *              table override mechanism is called to allow the host
225  *              OS to replace any table before it is installed in the root
226  *              table array.
227  *
228  ******************************************************************************/

230 void
231 AcpiTbInstallTable (
232     ACPI_PHYSICAL_ADDRESS   Address,
233     char                    *Signature,
234     UINT32                  TableIndex)
235 {
236     ACPI_TABLE_HEADER       *Table;
237     ACPI_TABLE_HEADER       *FinalTable;
238     ACPI_TABLE_DESC         *TableDesc;


241     if (!Address)
242     {
243         ACPI_ERROR ((AE_INFO, "Null physical address for ACPI table [%s]",
244             Signature));
245         return;
246     }

248     /* Map just the table header */

250     Table = AcpiOsMapMemory (Address, sizeof (ACPI_TABLE_HEADER));
251     if (!Table)
252     {
253         ACPI_ERROR ((AE_INFO, "Could not map memory for table [%s] at %p",
254             Signature, ACPI_CAST_PTR (void, Address)));
255         return;
256     }
```

```
258     /* If a particular signature is expected (DSDT/FACS), it must match */
259
260     if (Signature &&
261         !ACPI_COMPARE_NAME (Table->Signature, Signature))
262     {
263         ACPI_BIOS_ERROR ((AE_INFO,
264             "Invalid signature 0x%X for ACPI table, expected [%s]",
265             *ACPI_CAST_PTR (UINT32, Table->Signature), Signature));
266         goto UnmapAndExit;
267     }
268
269     /*
270      * Initialize the table entry. Set the pointer to NULL, since the
271      * table is not fully mapped at this time.
272      */
273     TableDesc = &AcpiGbl_RootTableList.Tables[TableIndex];
274
275     TableDesc->Address = Address;
276     TableDesc->Pointer = NULL;
277     TableDesc->Length = Table->Length;
278     TableDesc->Flags = ACPI_TABLE_ORIGIN_MAPPED;
279     ACPI_MOVE_32_TO_32 (TableDesc->Signature.Ascii, Table->Signature);
280
281     /*
282      * ACPI Table Override:
283      *
284      * Before we install the table, let the host OS override it with a new
285      * one if desired. Any table within the RSDT/XSDT can be replaced,
286      * including the DSDT which is pointed to by the FADT.
287      *
288      * NOTE: If the table is overridden, then FinalTable will contain a
289      * mapped pointer to the full new table. If the table is not overridden,
290      * or if there has been a physical override, then the table will be
291      * fully mapped later (in verify table). In any case, we must
292      * unmap the header that was mapped above.
293      */
294     FinalTable = AcpiTbTableOverride (Table, TableDesc);
295     if (!FinalTable)
296     {
297         FinalTable = Table; /* There was no override */
298     }
299
300     AcpiTbPrintTableHeader (TableDesc->Address, FinalTable);
301
302     /* Set the global integer width (based upon revision of the DSDT) */
303
304     if (TableIndex == ACPI_TABLE_INDEX_DSDT)
305     {
306         AcpiUtSetIntegerWidth (FinalTable->Revision);
307     }
308
309     /*
310      * If we have a physical override during this early loading of the ACPI
311      * tables, unmap the table for now. It will be mapped again later when
312      * it is actually used. This supports very early loading of ACPI tables,
313      * before virtual memory is fully initialized and running within the
314      * host OS. Note: A logical override has the ACPI_TABLE_ORIGIN_OVERRIDE
315      * flag set and will not be deleted below.
316      */
317     if (FinalTable != Table)
318     {
319         AcpiTbDeleteTable (TableDesc);
320     }
321
322
323 UnmapAndExit:
```



```
325     /* Always unmap the table header that we mapped above */
326
327     AcpiOsUnmapMemory (Table, sizeof (ACPI_TABLE_HEADER));
328 }
329
330
331 /*******************************************************************************
332  *
333  * FUNCTION:    AcpiTbGetRootTableEntry
334  *
335  * PARAMETERS:  TableEntry          - Pointer to the RSDT/XSDT table entry
336  *              TableEntrySize      - sizeof 32 or 64 (RSDT or XSDT)
337  *
338  * RETURN:      Physical address extracted from the root table
339  *
340  * DESCRIPTION: Get one root table entry. Handles 32-bit and 64-bit cases on
341  *              both 32-bit and 64-bit platforms
342  *
343  * NOTE:        ACPI_PHYSICAL_ADDRESS is 32-bit on 32-bit platforms, 64-bit on
344  *              64-bit platforms.
345  *
346  ******************************************************************************/
347
348 static ACPI_PHYSICAL_ADDRESS
349 AcpiTbGetRootTableEntry (
350     UINT8                   *TableEntry,
351     UINT32                  TableEntrySize)
352 {
353     UINT64                  Address64;
354
355
356     /*
357      * Get the table physical address (32-bit for RSDT, 64-bit for XSDT):
358      * Note: Addresses are 32-bit aligned (not 64) in both RSDT and XSDT
359      */
360     if (TableEntrySize == ACPI_RSDT_ENTRY_SIZE)
361     {
362         /*
363          * 32-bit platform, RSDT: Return 32-bit table entry
364          * 64-bit platform, RSDT: Expand 32-bit to 64-bit and return
365          */
366         return ((ACPI_PHYSICAL_ADDRESS) (*ACPI_CAST_PTR (UINT32, TableEntry)));
367     }
368     else
369     {
370         /*
371          * 32-bit platform, XSDT: Truncate 64-bit to 32-bit and return
372          * 64-bit platform, XSDT: Move (unaligned) 64-bit to local,
373          *  return 64-bit
374          */
375         ACPI_MOVE_64_TO_64 (&Address64, TableEntry);
376
377 #if ACPI_MACHINE_WIDTH == 32
378         if (Address64 > ACPI_UINT32_MAX)
379         {
380             /* Will truncate 64-bit address to 32 bits, issue warning */
381
382             ACPI_BIOS_WARNING ((AE_INFO,
383                 "64-bit Physical Address in XSDT is too large (0x%8.8X%8.8X),"
384                 " truncating",
385                 ACPI_FORMAT_UINT64 (Address64)));
386         }
387 #endif
388         return ((ACPI_PHYSICAL_ADDRESS) (Address64));
389     }
```

```
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiTbValidateXsdt
 *
 * PARAMETERS:  Address             - Physical address of the XSDT (from RSDP)
 *
 * RETURN:      Status. AE_OK if the table appears to be valid.
 *
 * DESCRIPTION: Validate an XSDT to ensure that it is of minimum size and does
 *              not contain any NULL entries. A problem that is seen in the
 *              field is that the XSDT exists, but is actually useless because
 *              of one or more (or all) NULL entries.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiTbValidateXsdt (
    ACPI_PHYSICAL_ADDRESS   XsdtAddress)
{
    ACPI_TABLE_HEADER       *Table;
    UINT8                   *NextEntry;
    ACPI_PHYSICAL_ADDRESS   Address;
    UINT32                  Length;
    UINT32                  EntryCount;
    ACPI_STATUS             Status;
    UINT32                  i;


    /* Get the XSDT length */

    Table = AcpiOsMapMemory (XsdtAddress, sizeof (ACPI_TABLE_HEADER));
    if (!Table)
    {
        return (AE_NO_MEMORY);
    }

    Length = Table->Length;
    AcpiOsUnmapMemory (Table, sizeof (ACPI_TABLE_HEADER));

    /*
     * Minimum XSDT length is the size of the standard ACPI header
     * plus one physical address entry
     */
    if (Length < (sizeof (ACPI_TABLE_HEADER) + ACPI_XSDT_ENTRY_SIZE))
    {
        return (AE_INVALID_TABLE_LENGTH);
    }

    /* Map the entire XSDT */

    Table = AcpiOsMapMemory (XsdtAddress, Length);
    if (!Table)
    {
        return (AE_NO_MEMORY);
    }

    /* Get the number of entries and pointer to first entry */

    Status = AE_OK;
    NextEntry = ACPI_ADD_PTR (UINT8, Table, sizeof (ACPI_TABLE_HEADER));
    EntryCount = (UINT32) ((Table->Length - sizeof (ACPI_TABLE_HEADER)) /
        ACPI_XSDT_ENTRY_SIZE);
```



```
    /* Validate each entry (physical address) within the XSDT */

    for (i = 0; i < EntryCount; i++)
    {
        Address = AcpiTbGetRootTableEntry (NextEntry, ACPI_XSDT_ENTRY_SIZE);
        if (!Address)
        {
            /* Detected a NULL entry, XSDT is invalid */

            Status = AE_NULL_ENTRY;
            break;
        }

        NextEntry += ACPI_XSDT_ENTRY_SIZE;
    }

    /* Unmap table */

    AcpiOsUnmapMemory (Table, Length);
    return (Status);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiTbParseRootTable
 *
 * PARAMETERS:  Rsdp                    - Pointer to the RSDP
 *
 * RETURN:      Status
 *
 * DESCRIPTION: This function is called to parse the Root System Description
 *              Table (RSDT or XSDT)
 *
 * NOTE:        Tables are mapped (not copied) for efficiency. The FACS must
 *              be mapped and cannot be copied because it contains the actual
 *              memory location of the ACPI Global Lock.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiTbParseRootTable (
    ACPI_PHYSICAL_ADDRESS   RsdpAddress)
{
    ACPI_TABLE_RSDP         *Rsdp;
    UINT32                  TableEntrySize;
    UINT32                  i;
    UINT32                  TableCount;
    ACPI_TABLE_HEADER       *Table;
    ACPI_PHYSICAL_ADDRESS   Address;
    UINT32                  Length;
    UINT8                   *TableEntry;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (TbParseRootTable);


    /* Map the entire RSDP and extract the address of the RSDT or XSDT */

    Rsdp = AcpiOsMapMemory (RsdpAddress, sizeof (ACPI_TABLE_RSDP));
    if (!Rsdp)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }
```

```
522     AcpiTbPrintTableHeader (RsdpAddress,
523         ACPI_CAST_PTR (ACPI_TABLE_HEADER, Rsdp));
524
525     /* Use XSDT if present and not overridden. Otherwise, use RSDT */
526
527     if ((Rsdp->Revision > 1) &&
528         Rsdp->XsdtPhysicalAddress &&
529         !AcpiGbl_DoNotUseXsdt)
530     {
531         /*
532          * RSDP contains an XSDT (64-bit physical addresses). We must use
533          * the XSDT if the revision is > 1 and the XSDT pointer is present,
534          * as per the ACPI specification.
535          */
536         Address = (ACPI_PHYSICAL_ADDRESS) Rsdp->XsdtPhysicalAddress;
537         TableEntrySize = ACPI_XSDT_ENTRY_SIZE;
538     }
539     else
540     {
541         /* Root table is an RSDT (32-bit physical addresses) */
542
543         Address = (ACPI_PHYSICAL_ADDRESS) Rsdp->RsdtPhysicalAddress;
544         TableEntrySize = ACPI_RSDT_ENTRY_SIZE;
545     }
546
547     /*
548      * It is not possible to map more than one entry in some environments,
549      * so unmap the RSDP here before mapping other tables
550      */
551     AcpiOsUnmapMemory (Rsdp, sizeof (ACPI_TABLE_RSDP));
552
553     /*
554      * If it is present and used, validate the XSDT for access/size
555      * and ensure that all table entries are at least non-NULL
556      */
557     if (TableEntrySize == ACPI_XSDT_ENTRY_SIZE)
558     {
559         Status = AcpiTbValidateXsdt (Address);
560         if (ACPI_FAILURE (Status))
561         {
562             ACPI_BIOS_WARNING ((AE_INFO, "XSDT is invalid (%s), using RSDT",
563                 AcpiFormatException (Status)));
564
565             /* Fall back to the RSDT */
566
567             Address = (ACPI_PHYSICAL_ADDRESS) Rsdp->RsdtPhysicalAddress;
568             TableEntrySize = ACPI_RSDT_ENTRY_SIZE;
569         }
570     }
571
572     /* Map the RSDT/XSDT table header to get the full table length */
573
574     Table = AcpiOsMapMemory (Address, sizeof (ACPI_TABLE_HEADER));
575     if (!Table)
576     {
577         return_ACPI_STATUS (AE_NO_MEMORY);
578     }
579
580     AcpiTbPrintTableHeader (Address, Table);
581
582     /*
583      * Validate length of the table, and map entire table.
584      * Minimum length table must contain at least one entry.
585      */
586     Length = Table->Length;
587     AcpiOsUnmapMemory (Table, sizeof (ACPI_TABLE_HEADER));
```



```
588
589     if (Length < (sizeof (ACPI_TABLE_HEADER) + TableEntrySize))
590     {
591         ACPI_BIOS_ERROR ((AE_INFO,
592             "Invalid table length 0x%X in RSDT/XSDT", Length));
593         return_ACPI_STATUS (AE_INVALID_TABLE_LENGTH);
594     }
595
596     Table = AcpiOsMapMemory (Address, Length);
597     if (!Table)
598     {
599         return_ACPI_STATUS (AE_NO_MEMORY);
600     }
601
602     /* Validate the root table checksum */
603
604     Status = AcpiTbVerifyChecksum (Table, Length);
605     if (ACPI_FAILURE (Status))
606     {
607         AcpiOsUnmapMemory (Table, Length);
608         return_ACPI_STATUS (Status);
609     }
610
611     /* Get the number of entries and pointer to first entry */
612
613     TableCount = (UINT32) ((Table->Length - sizeof (ACPI_TABLE_HEADER)) /
614         TableEntrySize);
615     TableEntry = ACPI_ADD_PTR (UINT8, Table, sizeof (ACPI_TABLE_HEADER));
616
617     /*
618      * First two entries in the table array are reserved for the DSDT
619      * and FACS, which are not actually present in the RSDT/XSDT - they
620      * come from the FADT
621      */
622     AcpiGbl_RootTableList.CurrentTableCount = 2;
623
624     /* Initialize the root table array from the RSDT/XSDT */
625
626     for (i = 0; i < TableCount; i++)
627     {
628         if (AcpiGbl_RootTableList.CurrentTableCount >=
629             AcpiGbl_RootTableList.MaxTableCount)
630         {
631             /* There is no more room in the root table array, attempt resize */
632
633             Status = AcpiTbResizeRootTableList ();
634             if (ACPI_FAILURE (Status))
635             {
636                 ACPI_WARNING ((AE_INFO, "Truncating %u table entries!",
637                     (unsigned) (TableCount -
638                     (AcpiGbl_RootTableList.CurrentTableCount - 2))));
639                 break;
640             }
641         }
642
643         /* Get the table physical address (32-bit for RSDT, 64-bit for XSDT) */
644
645         AcpiGbl_RootTableList.Tables[AcpiGbl_RootTableList.CurrentTableCount].Ad
646             AcpiTbGetRootTableEntry (TableEntry, TableEntrySize);
647
648         TableEntry += TableEntrySize;
649         AcpiGbl_RootTableList.CurrentTableCount++;
650     }
651
652     /*
653      * It is not possible to map more than one entry in some environments,
```

```
654      * so unmap the root table here before mapping other tables
655      */
656     AcpiOsUnmapMemory (Table, Length);

658     /*
659      * Complete the initialization of the root table array by examining
660      * the header of each table
661      */
662     for (i = 2; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
663     {
664         AcpiTbInstallTable (AcpiGbl_RootTableList.Tables[i].Address,
665             NULL, i);

667         /* Special case for FADT - validate it then get the DSDT and FACS */

669         if (ACPI_COMPARE_NAME (
670                 &AcpiGbl_RootTableList.Tables[i].Signature, ACPI_SIG_FADT))
671         {
672             AcpiTbParseFadt (i);
673         }
674     }

676     return_ACPI_STATUS (AE_OK);
677 }
```

```
**********************************************************
   15006 Thu Dec 26 13:49:34 2013
new/usr/src/common/acpica/components/tables/tbxface.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: tbxface - ACPI table-oriented external interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __TBXFACE_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#define _COMPONENT          ACPI_TABLES
        ACPI_MODULE_NAME    ("tbxface")


/*******************************************************************************
 *
 * FUNCTION:    AcpiAllocateRootTable
 *
 * PARAMETERS:  InitialTableCount   - Size of InitialTableArray, in number of
```



```
 *                                    ACPI_TABLE_DESC structures
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Allocate a root table array. Used by iASL compiler and
 *              AcpiInitializeTables.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiAllocateRootTable (
    UINT32                  InitialTableCount)
{

    AcpiGbl_RootTableList.MaxTableCount = InitialTableCount;
    AcpiGbl_RootTableList.Flags = ACPI_ROOT_ALLOW_RESIZE;

    return (AcpiTbResizeRootTableList ());
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiInitializeTables
 *
 * PARAMETERS:  InitialTableArray   - Pointer to an array of pre-allocated
 *                                    ACPI_TABLE_DESC structures. If NULL, the
 *                                    array is dynamically allocated.
 *              InitialTableCount   - Size of InitialTableArray, in number of
 *                                    ACPI_TABLE_DESC structures
 *              AllowResize         - Flag to tell Table Manager if resize of
 *                                    pre-allocated array is allowed. Ignored
 *                                    if InitialTableArray is NULL.
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize the table manager, get the RSDP and RSDT/XSDT.
 *
 * NOTE:        Allows static allocation of the initial table array in order
 *              to avoid the use of dynamic memory in confined environments
 *              such as the kernel boot sequence where it may not be available.
 *
 *              If the host OS memory managers are initialized, use NULL for
 *              InitialTableArray, and the table will be dynamically allocated.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiInitializeTables (
    ACPI_TABLE_DESC         *InitialTableArray,
    UINT32                  InitialTableCount,
    BOOLEAN                 AllowResize)
{
    ACPI_PHYSICAL_ADDRESS   RsdpAddress;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiInitializeTables);


    /*
     * Setup the Root Table Array and allocate the table array
     * if requested
     */
    if (!InitialTableArray)
    {
```

```
126         Status = AcpiAllocateRootTable (InitialTableCount);
127         if (ACPI_FAILURE (Status))
128         {
129             return_ACPI_STATUS (Status);
130         }
131     }
132     else
133     {
134         /* Root Table Array has been statically allocated by the host */

136         ACPI_MEMSET (InitialTableArray, 0,
137             (ACPI_SIZE) InitialTableCount * sizeof (ACPI_TABLE_DESC));

139         AcpiGbl_RootTableList.Tables = InitialTableArray;
140         AcpiGbl_RootTableList.MaxTableCount = InitialTableCount;
141         AcpiGbl_RootTableList.Flags = ACPI_ROOT_ORIGIN_UNKNOWN;
142         if (AllowResize)
143         {
144             AcpiGbl_RootTableList.Flags |= ACPI_ROOT_ALLOW_RESIZE;
145         }
146     }

148     /* Get the address of the RSDP */

150     RsdpAddress = AcpiOsGetRootPointer ();
151     if (!RsdpAddress)
152     {
153         return_ACPI_STATUS (AE_NOT_FOUND);
154     }

156     /*
157      * Get the root table (RSDT or XSDT) and extract all entries to the local
158      * Root Table Array. This array contains the information of the RSDT/XSDT
159      * in a common, more useable format.
160      */
161     Status = AcpiTbParseRootTable (RsdpAddress);
162     return_ACPI_STATUS (Status);
163 }

165 ACPI_EXPORT_SYMBOL_INIT (AcpiInitializeTables)


168 /*******************************************************************************
169  *
170  * FUNCTION:    AcpiReallocateRootTable
171  *
172  * PARAMETERS:  None
173  *
174  * RETURN:      Status
175  *
176  * DESCRIPTION: Reallocate Root Table List into dynamic memory. Copies the
177  *              root list from the previously provided scratch area. Should
178  *              be called once dynamic memory allocation is available in the
179  *              kernel.
180  *
181  ******************************************************************************/

183 ACPI_STATUS
184 AcpiReallocateRootTable (
185     void)
186 {
187     ACPI_STATUS             Status;


190     ACPI_FUNCTION_TRACE (AcpiReallocateRootTable);
```



```
193     /*
194      * Only reallocate the root table if the host provided a static buffer
195      * for the table array in the call to AcpiInitializeTables.
196      */
197     if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
198     {
199         return_ACPI_STATUS (AE_SUPPORT);
200     }

202     AcpiGbl_RootTableList.Flags |= ACPI_ROOT_ALLOW_RESIZE;

204     Status = AcpiTbResizeRootTableList ();
205     return_ACPI_STATUS (Status);
206 }

208 ACPI_EXPORT_SYMBOL_INIT (AcpiReallocateRootTable)


211 /*******************************************************************************
212  *
213  * FUNCTION:    AcpiGetTableHeader
214  *
215  * PARAMETERS:  Signature           - ACPI signature of needed table
216  *              Instance            - Which instance (for SSDTs)
217  *              OutTableHeader      - The pointer to the table header to fill
218  *
219  * RETURN:      Status and pointer to mapped table header
220  *
221  * DESCRIPTION: Finds an ACPI table header.
222  *
223  * NOTE:        Caller is responsible in unmapping the header with
224  *              AcpiOsUnmapMemory
225  *
226  ******************************************************************************/

228 ACPI_STATUS
229 AcpiGetTableHeader (
230     char                    *Signature,
231     UINT32                  Instance,
232     ACPI_TABLE_HEADER       *OutTableHeader)
233 {
234     UINT32                  i;
235     UINT32                  j;
236     ACPI_TABLE_HEADER       *Header;


239     /* Parameter validation */

241     if (!Signature || !OutTableHeader)
242     {
243         return (AE_BAD_PARAMETER);
244     }

246     /* Walk the root table list */

248     for (i = 0, j = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
249     {
250         if (!ACPI_COMPARE_NAME (&(AcpiGbl_RootTableList.Tables[i].Signature),
251                     Signature))
252         {
253             continue;
254         }

256         if (++j < Instance)
257         {
```

```
258             continue;
259         }

261         if (!AcpiGbl_RootTableList.Tables[i].Pointer)
262         {
263             if ((AcpiGbl_RootTableList.Tables[i].Flags &
264                     ACPI_TABLE_ORIGIN_MASK) ==
265                 ACPI_TABLE_ORIGIN_MAPPED)
266             {
267                 Header = AcpiOsMapMemory (
268                             AcpiGbl_RootTableList.Tables[i].Address,
269                             sizeof (ACPI_TABLE_HEADER));
270                 if (!Header)
271                 {
272                     return (AE_NO_MEMORY);
273                 }

275                 ACPI_MEMCPY (OutTableHeader, Header,
276                     sizeof (ACPI_TABLE_HEADER));
277                 AcpiOsUnmapMemory (Header, sizeof (ACPI_TABLE_HEADER));
278             }
279             else
280             {
281                 return (AE_NOT_FOUND);
282             }
283         }
284         else
285         {
286             ACPI_MEMCPY (OutTableHeader,
287                 AcpiGbl_RootTableList.Tables[i].Pointer,
288                 sizeof (ACPI_TABLE_HEADER));
289         }

291         return (AE_OK);
292     }

294     return (AE_NOT_FOUND);
295 }

297 ACPI_EXPORT_SYMBOL (AcpiGetTableHeader)


300 /*******************************************************************************
301  *
302  * FUNCTION:    AcpiGetTable
303  *
304  * PARAMETERS:  Signature           - ACPI signature of needed table
305  *              Instance            - Which instance (for SSDTs)
306  *              OutTable            - Where the pointer to the table is returned
307  *
308  * RETURN:      Status and pointer to the requested table
309  *
310  * DESCRIPTION: Finds and verifies an ACPI table. Table must be in the
311  *              RSDT/XSDT.
312  *
313  ******************************************************************************/

315 ACPI_STATUS
316 AcpiGetTable (
317     char                    *Signature,
318     UINT32                  Instance,
319     ACPI_TABLE_HEADER       **OutTable)
320 {
321     UINT32                  i;
322     UINT32                  j;
323     ACPI_STATUS             Status;
```



```


326     /* Parameter validation */

328     if (!Signature || !OutTable)
329     {
330         return (AE_BAD_PARAMETER);
331     }

333     /* Walk the root table list */

335     for (i = 0, j = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
336     {
337         if (!ACPI_COMPARE_NAME (&(AcpiGbl_RootTableList.Tables[i].Signature),
338                 Signature))
339         {
340             continue;
341         }

343         if (++j < Instance)
344         {
345             continue;
346         }

348         Status = AcpiTbVerifyTable (&AcpiGbl_RootTableList.Tables[i]);
349         if (ACPI_SUCCESS (Status))
350         {
351             *OutTable = AcpiGbl_RootTableList.Tables[i].Pointer;
352         }

354         return (Status);
355     }

357     return (AE_NOT_FOUND);
358 }

360 ACPI_EXPORT_SYMBOL (AcpiGetTable)


363 /*******************************************************************************
364  *
365  * FUNCTION:    AcpiGetTableByIndex
366  *
367  * PARAMETERS:  TableIndex          - Table index
368  *              Table               - Where the pointer to the table is returned
369  *
370  * RETURN:      Status and pointer to the requested table
371  *
372  * DESCRIPTION: Obtain a table by an index into the global table list. Used
373  *              internally also.
374  *
375  ******************************************************************************/

377 ACPI_STATUS
378 AcpiGetTableByIndex (
379     UINT32                  TableIndex,
380     ACPI_TABLE_HEADER       **Table)
381 {
382     ACPI_STATUS             Status;


385     ACPI_FUNCTION_TRACE (AcpiGetTableByIndex);


388     /* Parameter validation */
```

```
390     if (!Table)
391     {
392         return_ACPI_STATUS (AE_BAD_PARAMETER);
393     }

395     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);

397     /* Validate index */

399     if (TableIndex >= AcpiGbl_RootTableList.CurrentTableCount)
400     {
401         (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
402         return_ACPI_STATUS (AE_BAD_PARAMETER);
403     }

405     if (!AcpiGbl_RootTableList.Tables[TableIndex].Pointer)
406     {
407         /* Table is not mapped, map it */

409         Status = AcpiTbVerifyTable (&AcpiGbl_RootTableList.Tables[TableIndex]);
410         if (ACPI_FAILURE (Status))
411         {
412             (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
413             return_ACPI_STATUS (Status);
414         }
415     }

417     *Table = AcpiGbl_RootTableList.Tables[TableIndex].Pointer;
418     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
419     return_ACPI_STATUS (AE_OK);
420 }

422 ACPI_EXPORT_SYMBOL (AcpiGetTableByIndex)


425 /*******************************************************************************
426  *
427  * FUNCTION:    AcpiInstallTableHandler
428  *
429  * PARAMETERS:  Handler         - Table event handler
430  *              Context         - Value passed to the handler on each event
431  *
432  * RETURN:      Status
433  *
434  * DESCRIPTION: Install a global table event handler.
435  *
436  ******************************************************************************/

438 ACPI_STATUS
439 AcpiInstallTableHandler (
440     ACPI_TABLE_HANDLER      Handler,
441     void                    *Context)
442 {
443     ACPI_STATUS             Status;


446     ACPI_FUNCTION_TRACE (AcpiInstallTableHandler);


449     if (!Handler)
450     {
451         return_ACPI_STATUS (AE_BAD_PARAMETER);
452     }

454     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
455     if (ACPI_FAILURE (Status))
```



```
456     {
457         return_ACPI_STATUS (Status);
458     }

460     /* Don't allow more than one handler */

462     if (AcpiGbl_TableHandler)
463     {
464         Status = AE_ALREADY_EXISTS;
465         goto Cleanup;
466     }

468     /* Install the handler */

470     AcpiGbl_TableHandler = Handler;
471     AcpiGbl_TableHandlerContext = Context;

473 Cleanup:
474     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
475     return_ACPI_STATUS (Status);
476 }

478 ACPI_EXPORT_SYMBOL (AcpiInstallTableHandler)


481 /*******************************************************************************
482  *
483  * FUNCTION:    AcpiRemoveTableHandler
484  *
485  * PARAMETERS:  Handler         - Table event handler that was installed
486  *                                previously.
487  *
488  * RETURN:      Status
489  *
490  * DESCRIPTION: Remove a table event handler
491  *
492  ******************************************************************************/

494 ACPI_STATUS
495 AcpiRemoveTableHandler (
496     ACPI_TABLE_HANDLER      Handler)
497 {
498     ACPI_STATUS             Status;


501     ACPI_FUNCTION_TRACE (AcpiRemoveTableHandler);


504     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
505     if (ACPI_FAILURE (Status))
506     {
507         return_ACPI_STATUS (Status);
508     }

510     /* Make sure that the installed handler is the same */

512     if (!Handler ||
513         Handler != AcpiGbl_TableHandler)
514     {
515         Status = AE_BAD_PARAMETER;
516         goto Cleanup;
517     }

519     /* Remove the handler */

521     AcpiGbl_TableHandler = NULL;
```

```
523 Cleanup:
524     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
525     return_ACPI_STATUS (Status);
526 }

528 ACPI_EXPORT_SYMBOL (AcpiRemoveTableHandler)
```

```
**********************************************************
   12699 Thu Dec 26 13:49:34 2013
new/usr/src/common/acpica/components/tables/tbxfload.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: tbxfload - Table load/unload external interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __TBXFLOAD_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "actables.h"

#define _COMPONENT          ACPI_TABLES
        ACPI_MODULE_NAME    ("tbxfload")

/* Local prototypes */

static ACPI_STATUS
AcpiTbLoadNamespace (
    void);
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiLoadTables
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Load the ACPI tables from the RSDT/XSDT
 *
 *****************************************************************************/

ACPI_STATUS
AcpiLoadTables (
    void)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiLoadTables);


    /* Load the namespace from the tables */

    Status = AcpiTbLoadNamespace ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status,
            "While loading namespace from ACPI tables"));
    }

    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL_INIT (AcpiLoadTables)


/******************************************************************************
 *
 * FUNCTION:    AcpiTbLoadNamespace
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Load the namespace from the DSDT and all SSDTs/PSDTs found in
 *              the RSDT/XSDT.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiTbLoadNamespace (
    void)
{
    ACPI_STATUS             Status;
    UINT32                  i;
    ACPI_TABLE_HEADER       *NewDsdt;


    ACPI_FUNCTION_TRACE (TbLoadNamespace);


    (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);

    /*
```

```
127      * Load the namespace. The DSDT is required, but any SSDT and
128      * PSDT tables are optional. Verify the DSDT.
129      */
130     if (!AcpiGbl_RootTableList.CurrentTableCount ||
131         !ACPI_COMPARE_NAME (
132             &(AcpiGbl_RootTableList.Tables[ACPI_TABLE_INDEX_DSDT].Signature),
133             ACPI_SIG_DSDT) ||
134          ACPI_FAILURE (AcpiTbVerifyTable (
135             &AcpiGbl_RootTableList.Tables[ACPI_TABLE_INDEX_DSDT])))
136     {
137         Status = AE_NO_ACPI_TABLES;
138         goto UnlockAndExit;
139     }
140
141     /*
142      * Save the DSDT pointer for simple access. This is the mapped memory
143      * address. We must take care here because the address of the .Tables
144      * array can change dynamically as tables are loaded at run-time. Note:
145      * .Pointer field is not validated until after call to AcpiTbVerifyTable.
146      */
147     AcpiGbl_DSDT = AcpiGbl_RootTableList.Tables[ACPI_TABLE_INDEX_DSDT].Pointer;
148
149     /*
150      * Optionally copy the entire DSDT to local memory (instead of simply
151      * mapping it.) There are some BIOSs that corrupt or replace the original
152      * DSDT, creating the need for this option. Default is FALSE, do not copy
153      * the DSDT.
154      */
155     if (AcpiGbl_CopyDsdtLocally)
156     {
157         NewDsdt = AcpiTbCopyDsdt (ACPI_TABLE_INDEX_DSDT);
158         if (NewDsdt)
159         {
160             AcpiGbl_DSDT = NewDsdt;
161         }
162     }
163
164     /*
165      * Save the original DSDT header for detection of table corruption
166      * and/or replacement of the DSDT from outside the OS.
167      */
168     ACPI_MEMCPY (&AcpiGbl_OriginalDsdtHeader, AcpiGbl_DSDT,
169         sizeof (ACPI_TABLE_HEADER));
170
171     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
172
173     /* Load and parse tables */
174
175     Status = AcpiNsLoadTable (ACPI_TABLE_INDEX_DSDT, AcpiGbl_RootNode);
176     if (ACPI_FAILURE (Status))
177     {
178         return_ACPI_STATUS (Status);
179     }
180
181     /* Load any SSDT or PSDT tables. Note: Loop leaves tables locked */
182
183     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);
184     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; ++i)
185     {
186         if ((!ACPI_COMPARE_NAME (&(AcpiGbl_RootTableList.Tables[i].Signature),
187                     ACPI_SIG_SSDT) &&
188              !ACPI_COMPARE_NAME (&(AcpiGbl_RootTableList.Tables[i].Signature),
189                     ACPI_SIG_PSDT)) ||
190             ACPI_FAILURE (AcpiTbVerifyTable (
191                &AcpiGbl_RootTableList.Tables[i])))
192         {
```



```
193             continue;
194         }
195
196         /*
197          * Optionally do not load any SSDTs from the RSDT/XSDT. This can
198          * be useful for debugging ACPI problems on some machines.
199          */
200         if (AcpiGbl_DisableSsdtTableLoad)
201         {
202             ACPI_INFO ((AE_INFO, "Ignoring %4.4s at %p",
203                 AcpiGbl_RootTableList.Tables[i].Signature.Ascii,
204                 ACPI_CAST_PTR (void, AcpiGbl_RootTableList.Tables[i].Address)));
205             continue;
206         }
207
208         /* Ignore errors while loading tables, get as many as possible */
209
210         (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
211         (void) AcpiNsLoadTable (i, AcpiGbl_RootNode);
212         (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);
213     }
214
215     ACPI_INFO ((AE_INFO, "All ACPI Tables successfully acquired"));
216
217 UnlockAndExit:
218     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
219     return_ACPI_STATUS (Status);
220 }
221
222
223 /*******************************************************************************
224  *
225  * FUNCTION:    AcpiLoadTable
226  *
227  * PARAMETERS:  Table               - Pointer to a buffer containing the ACPI
228  *                                    table to be loaded.
229  *
230  * RETURN:      Status
231  *
232  * DESCRIPTION: Dynamically load an ACPI table from the caller's buffer. Must
233  *              be a valid ACPI table with a valid ACPI table header.
234  *              Note1: Mainly intended to support hotplug addition of SSDTs.
235  *              Note2: Does not copy the incoming table. User is responsible
236  *              to ensure that the table is not deleted or unmapped.
237  *
238  ******************************************************************************/
239
240 ACPI_STATUS
241 AcpiLoadTable (
242     ACPI_TABLE_HEADER       *Table)
243 {
244     ACPI_STATUS             Status;
245     ACPI_TABLE_DESC         TableDesc;
246     UINT32                  TableIndex;
247
248
249     ACPI_FUNCTION_TRACE (AcpiLoadTable);
250
251
252     /* Parameter validation */
253
254     if (!Table)
255     {
256         return_ACPI_STATUS (AE_BAD_PARAMETER);
257     }
```

```
259     /* Init local table descriptor */

261     ACPI_MEMSET (&TableDesc, 0, sizeof (ACPI_TABLE_DESC));
262     TableDesc.Address = ACPI_PTR_TO_PHYSADDR (Table);
263     TableDesc.Pointer = Table;
264     TableDesc.Length = Table->Length;
265     TableDesc.Flags = ACPI_TABLE_ORIGIN_UNKNOWN;

267     /* Must acquire the interpreter lock during this operation */

269     Status = AcpiUtAcquireMutex (ACPI_MTX_INTERPRETER);
270     if (ACPI_FAILURE (Status))
271     {
272         return_ACPI_STATUS (Status);
273     }

275     /* Install the table and load it into the namespace */

277     ACPI_INFO ((AE_INFO, "Host-directed Dynamic ACPI Table Load:"));
278     Status = AcpiTbAddTable (&TableDesc, &TableIndex);
279     if (ACPI_FAILURE (Status))
280     {
281         goto UnlockAndExit;
282     }

284     Status = AcpiNsLoadTable (TableIndex, AcpiGbl_RootNode);

286     /* Invoke table handler if present */

288     if (AcpiGbl_TableHandler)
289     {
290         (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_LOAD, Table,
291                     AcpiGbl_TableHandlerContext);
292     }

294 UnlockAndExit:
295     (void) AcpiUtReleaseMutex (ACPI_MTX_INTERPRETER);
296     return_ACPI_STATUS (Status);
297 }

299 ACPI_EXPORT_SYMBOL (AcpiLoadTable)


302 /*******************************************************************************
303  *
304  * FUNCTION:    AcpiUnloadParentTable
305  *
306  * PARAMETERS:  Object              - Handle to any namespace object owned by
307  *                                    the table to be unloaded
308  *
309  * RETURN:      Status
310  *
311  * DESCRIPTION: Via any namespace object within an SSDT or OEMx table, unloads
312  *              the table and deletes all namespace objects associated with
313  *              that table. Unloading of the DSDT is not allowed.
314  *              Note: Mainly intended to support hotplug removal of SSDTs.
315  *
316  ******************************************************************************/

318 ACPI_STATUS
319 AcpiUnloadParentTable (
320     ACPI_HANDLE             Object)
321 {
322     ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Object);
323     ACPI_STATUS             Status = AE_NOT_EXIST;
324     ACPI_OWNER_ID           OwnerId;
```



```
325     UINT32                  i;


328     ACPI_FUNCTION_TRACE (AcpiUnloadParentTable);


331     /* Parameter validation */

333     if (!Object)
334     {
335         return_ACPI_STATUS (AE_BAD_PARAMETER);
336     }

338     /*
339      * The node OwnerId is currently the same as the parent table ID.
340      * However, this could change in the future.
341      */
342     OwnerId = Node->OwnerId;
343     if (!OwnerId)
344     {
345         /* OwnerId==0 means DSDT is the owner. DSDT cannot be unloaded */

347         return_ACPI_STATUS (AE_TYPE);
348     }

350     /* Must acquire the interpreter lock during this operation */

352     Status = AcpiUtAcquireMutex (ACPI_MTX_INTERPRETER);
353     if (ACPI_FAILURE (Status))
354     {
355         return_ACPI_STATUS (Status);
356     }

358     /* Find the table in the global table list */

360     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
361     {
362         if (OwnerId != AcpiGbl_RootTableList.Tables[i].OwnerId)
363         {
364             continue;
365         }

367         /*
368          * Allow unload of SSDT and OEMx tables only. Do not allow unload
369          * of the DSDT. No other types of tables should get here, since
370          * only these types can contain AML and thus are the only types
371          * that can create namespace objects.
372          */
373         if (ACPI_COMPARE_NAME (
374             AcpiGbl_RootTableList.Tables[i].Signature.Ascii,
375             ACPI_SIG_DSDT))
376         {
377             Status = AE_TYPE;
378             break;
379         }

381         /* Ensure the table is actually loaded */

383         if (!AcpiTbIsTableLoaded (i))
384         {
385             Status = AE_NOT_EXIST;
386             break;
387         }

389         /* Invoke table handler if present */
```

```
391         if (AcpiGbl_TableHandler)
392         {
393             (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_UNLOAD,
394                         AcpiGbl_RootTableList.Tables[i].Pointer,
395                         AcpiGbl_TableHandlerContext);
396         }

398         /*
399          * Delete all namespace objects owned by this table. Note that
400          * these objects can appear anywhere in the namespace by virtue
401          * of the AML "Scope" operator. Thus, we need to track ownership
402          * by an ID, not simply a position within the hierarchy.
403          */
404         Status = AcpiTbDeleteNamespaceByOwner (i);
405         if (ACPI_FAILURE (Status))
406         {
407             break;
408         }

410         Status = AcpiTbReleaseOwnerId (i);
411         AcpiTbSetTableLoadedFlag (i, FALSE);
412         break;
413     }

415     (void) AcpiUtReleaseMutex (ACPI_MTX_INTERPRETER);
416     return_ACPI_STATUS (Status);
417 }

419 ACPI_EXPORT_SYMBOL (AcpiUnloadParentTable)
```

```
**********************************************************
    8842 Thu Dec 26 13:49:35 2013
new/usr/src/common/acpica/components/tables/tbxfroot.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: tbxfroot - Find the root ACPI table (RSDT)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __TBXFROOT_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "actables.h"


  51 #define _COMPONENT          ACPI_TABLES
  52         ACPI_MODULE_NAME    ("tbxfroot")

  54 /* Local prototypes */

  56 static UINT8 *
  57 AcpiTbScanMemoryForRsdp (
  58     UINT8                   *StartAddress,
  59     UINT32                  Length);
```



```
  61 static ACPI_STATUS
  62 AcpiTbValidateRsdp (
  63     ACPI_TABLE_RSDP         *Rsdp);


  55 /******************************************************************************
  56  *
  57  * FUNCTION:    AcpiTbValidateRsdp
  58  *
  59  * PARAMETERS:  Rsdp                - Pointer to unvalidated RSDP
  60  *
  61  * RETURN:      Status
  62  *
  63  * DESCRIPTION: Validate the RSDP (ptr)
  64  *
  65  *****************************************************************************/

  67 ACPI_STATUS
  78 static ACPI_STATUS
  68 AcpiTbValidateRsdp (
  69     ACPI_TABLE_RSDP         *Rsdp)
  70 {
  82     ACPI_FUNCTION_ENTRY ();


  72     /*
  73      * The signature and checksum must both be correct
  74      *
  75      * Note: Sometimes there exists more than one RSDP in memory; the valid
  76      * RSDP has a valid checksum, all others have an invalid checksum.
  77      */
  78     if (!ACPI_VALIDATE_RSDP_SIG (Rsdp->Signature))
  91     if (ACPI_STRNCMP ((char *) Rsdp, ACPI_SIG_RSDP,
  92             sizeof (ACPI_SIG_RSDP)-1) != 0)
  79     {
  80         /* Nope, BAD Signature */

  82         return (AE_BAD_SIGNATURE);
  83     }

  85     /* Check the standard checksum */

  87     if (AcpiTbChecksum ((UINT8 *) Rsdp, ACPI_RSDP_CHECKSUM_LENGTH) != 0)
  88     {
  89         return (AE_BAD_CHECKSUM);
  90     }

  92     /* Check extended checksum if table version >= 2 */

  94     if ((Rsdp->Revision >= 2) &&
  95         (AcpiTbChecksum ((UINT8 *) Rsdp, ACPI_RSDP_XCHECKSUM_LENGTH) != 0))
  96     {
  97         return (AE_BAD_CHECKSUM);
  98     }

 100     return (AE_OK);
 101 }


 104 /******************************************************************************
 105  *
 106  * FUNCTION:    AcpiFindRootPointer
 107  *
 108  * PARAMETERS:  TableAddress            - Where the table pointer is returned
 109  *
```

```
 110  * RETURN:      Status, RSDP physical address
 111  *
 112  * DESCRIPTION: Search lower 1Mbyte of memory for the root system descriptor
 113  *              pointer structure. If it is found, set *RSDP to point to it.
 114  *
 115  * NOTE1:       The RSDP must be either in the first 1K of the Extended
 116  *              BIOS Data Area or between E0000 and FFFFF (From ACPI Spec.)
 117  *              Only a 32-bit physical address is necessary.
 118  *
 119  * NOTE2:       This function is always available, regardless of the
 120  *              initialization state of the rest of ACPI.
 121  *
 122  ******************************************************************************/

 124 ACPI_STATUS
 125 AcpiFindRootPointer (
 126     ACPI_SIZE               *TableAddress)
 127 {
 128     UINT8                   *TablePtr;
 129     UINT8                   *MemRover;
 130     UINT32                  PhysicalAddress;


 133     ACPI_FUNCTION_TRACE (AcpiFindRootPointer);


 136     /* 1a) Get the location of the Extended BIOS Data Area (EBDA) */

 138     TablePtr = AcpiOsMapMemory (
 139                 (ACPI_PHYSICAL_ADDRESS) ACPI_EBDA_PTR_LOCATION,
 140                 ACPI_EBDA_PTR_LENGTH);
 141     if (!TablePtr)
 142     {
 143         ACPI_ERROR ((AE_INFO,
 144             "Could not map memory at 0x%8.8X for length %u",
 145             ACPI_EBDA_PTR_LOCATION, ACPI_EBDA_PTR_LENGTH));

 147         return_ACPI_STATUS (AE_NO_MEMORY);
 148     }

 150     ACPI_MOVE_16_TO_32 (&PhysicalAddress, TablePtr);

 152     /* Convert segment part to physical address */

 154     PhysicalAddress <<= 4;
 155     AcpiOsUnmapMemory (TablePtr, ACPI_EBDA_PTR_LENGTH);

 157     /* EBDA present? */

 159     if (PhysicalAddress > 0x400)
 160     {
 161         /*
 162          * 1b) Search EBDA paragraphs (EBDA is required to be a
 163          *     minimum of 1K length)
 164          */
 165         TablePtr = AcpiOsMapMemory (
 166                     (ACPI_PHYSICAL_ADDRESS) PhysicalAddress,
 167                     ACPI_EBDA_WINDOW_SIZE);
 168         if (!TablePtr)
 169         {
 170             ACPI_ERROR ((AE_INFO,
 171                 "Could not map memory at 0x%8.8X for length %u",
 172                 PhysicalAddress, ACPI_EBDA_WINDOW_SIZE));

 174             return_ACPI_STATUS (AE_NO_MEMORY);
 175         }
```



```
 177         MemRover = AcpiTbScanMemoryForRsdp (TablePtr, ACPI_EBDA_WINDOW_SIZE);
 178         AcpiOsUnmapMemory (TablePtr, ACPI_EBDA_WINDOW_SIZE);

 180         if (MemRover)
 181         {
 182             /* Return the physical address */

 184             PhysicalAddress += (UINT32) ACPI_PTR_DIFF (MemRover, TablePtr);

 186             *TableAddress = PhysicalAddress;
 187             return_ACPI_STATUS (AE_OK);
 188         }
 189     }

 191     /*
 192      * 2) Search upper memory: 16-byte boundaries in E0000h-FFFFFh
 193      */
 194     TablePtr = AcpiOsMapMemory (
 195                 (ACPI_PHYSICAL_ADDRESS) ACPI_HI_RSDP_WINDOW_BASE,
 196                 ACPI_HI_RSDP_WINDOW_SIZE);

 198     if (!TablePtr)
 199     {
 200         ACPI_ERROR ((AE_INFO,
 201             "Could not map memory at 0x%8.8X for length %u",
 202             ACPI_HI_RSDP_WINDOW_BASE, ACPI_HI_RSDP_WINDOW_SIZE));

 204         return_ACPI_STATUS (AE_NO_MEMORY);
 205     }

 207     MemRover = AcpiTbScanMemoryForRsdp (TablePtr, ACPI_HI_RSDP_WINDOW_SIZE);
 208     AcpiOsUnmapMemory (TablePtr, ACPI_HI_RSDP_WINDOW_SIZE);

 210     if (MemRover)
 211     {
 212         /* Return the physical address */

 214         PhysicalAddress = (UINT32)
 215             (ACPI_HI_RSDP_WINDOW_BASE + ACPI_PTR_DIFF (MemRover, TablePtr));

 217         *TableAddress = PhysicalAddress;
 218         return_ACPI_STATUS (AE_OK);
 219     }

 221     /* A valid RSDP was not found */

 223     ACPI_BIOS_ERROR ((AE_INFO, "A valid RSDP was not found"));
 237     ACPI_ERROR ((AE_INFO, "A valid RSDP was not found"));
 224     return_ACPI_STATUS (AE_NOT_FOUND);
 225 }

 227 ACPI_EXPORT_SYMBOL (AcpiFindRootPointer)


 230 /*******************************************************************************
 231  *
 232  * FUNCTION:    AcpiTbScanMemoryForRsdp
 233  *
 234  * PARAMETERS:  StartAddress        - Starting pointer for search
 235  *              Length              - Maximum length to search
 236  *
 237  * RETURN:      Pointer to the RSDP if found, otherwise NULL.
 238  *
 239  * DESCRIPTION: Search a block of memory for the RSDP signature
 240  *
```

```
241  *****************************************************************************/

243 UINT8 *
257 static UINT8 *
244 AcpiTbScanMemoryForRsdp (
245     UINT8                   *StartAddress,
246     UINT32                  Length)
247 {
248     ACPI_STATUS             Status;
249     UINT8                   *MemRover;
250     UINT8                   *EndAddress;


253     ACPI_FUNCTION_TRACE (TbScanMemoryForRsdp);


256     EndAddress = StartAddress + Length;

258     /* Search from given start address for the requested length */

260     for (MemRover = StartAddress; MemRover < EndAddress;
261          MemRover += ACPI_RSDP_SCAN_STEP)
262     {
263         /* The RSDP signature and checksum must both be correct */

265         Status = AcpiTbValidateRsdp (ACPI_CAST_PTR (ACPI_TABLE_RSDP, MemRover));
266         if (ACPI_SUCCESS (Status))
267         {
268             /* Sig and checksum valid, we have found a real RSDP */

270             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
271                 "RSDP located at physical address %p\n", MemRover));
272             return_PTR (MemRover);
273         }

275         /* No sig match or bad checksum, keep searching */
276     }

278     /* Searched entire block, no RSDP was found */

280     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
281         "Searched entire block from %p, valid RSDP was not found\n",
282         StartAddress));
283     return_PTR (NULL);
284 }
```

```
**********************************************************
   10352 Thu Dec 26 13:49:35 2013
new/usr/src/common/acpica/components/utilities/utaddress.c
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utaddress - OpRegion address range check
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTADDRESS_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utaddress")


/******************************************************************************
 *
 * FUNCTION:    AcpiUtAddAddressRange
 *
 * PARAMETERS:  SpaceId             - Address space ID
 *              Address             - OpRegion start address
```



```
 *              Length              - OpRegion length
 *              RegionNode          - OpRegion namespace node
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Add the Operation Region address range to the global list.
 *              The only supported Space IDs are Memory and I/O. Called when
 *              the OpRegion address/length operands are fully evaluated.
 *
 * MUTEX:       Locks the namespace
 *
 * NOTE: Because this interface is only called when an OpRegion argument
 * list is evaluated, there cannot be any duplicate RegionNodes.
 * Duplicate Address/Length values are allowed, however, so that multiple
 * address conflicts can be detected.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtAddAddressRange (
    ACPI_ADR_SPACE_TYPE     SpaceId,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  Length,
    ACPI_NAMESPACE_NODE     *RegionNode)
{
    ACPI_ADDRESS_RANGE      *RangeInfo;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (UtAddAddressRange);


    if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
        (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
    {
        return_ACPI_STATUS (AE_OK);
    }

    /* Allocate/init a new info block, add it to the appropriate list */

    RangeInfo = ACPI_ALLOCATE (sizeof (ACPI_ADDRESS_RANGE));
    if (!RangeInfo)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    RangeInfo->StartAddress = Address;
    RangeInfo->EndAddress = (Address + Length - 1);
    RangeInfo->RegionNode = RegionNode;

    Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
    if (ACPI_FAILURE (Status))
    {
        ACPI_FREE (RangeInfo);
        return_ACPI_STATUS (Status);
    }

    RangeInfo->Next = AcpiGbl_AddressRangeList[SpaceId];
    AcpiGbl_AddressRangeList[SpaceId] = RangeInfo;

    ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
        "\nAdded [%4.4s] address range: 0x%p-0x%p\n",
        AcpiUtGetNodeName (RangeInfo->RegionNode),
        ACPI_CAST_PTR (void, Address),
        ACPI_CAST_PTR (void, RangeInfo->EndAddress)));
```

```
    (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
    return_ACPI_STATUS (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtRemoveAddressRange
 *
 * PARAMETERS:  SpaceId             - Address space ID
 *              RegionNode          - OpRegion namespace node
 *
 * RETURN:      None
 *
 * DESCRIPTION: Remove the Operation Region from the global list. The only
 *              supported Space IDs are Memory and I/O. Called when an
 *              OpRegion is deleted.
 *
 * MUTEX:       Assumes the namespace is locked
 *
 ******************************************************************************/

void
AcpiUtRemoveAddressRange (
    ACPI_ADR_SPACE_TYPE     SpaceId,
    ACPI_NAMESPACE_NODE     *RegionNode)
{
    ACPI_ADDRESS_RANGE      *RangeInfo;
    ACPI_ADDRESS_RANGE      *Prev;


    ACPI_FUNCTION_TRACE (UtRemoveAddressRange);


    if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
        (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
    {
        return_VOID;
    }

    /* Get the appropriate list head and check the list */

    RangeInfo = Prev = AcpiGbl_AddressRangeList[SpaceId];
    while (RangeInfo)
    {
        if (RangeInfo->RegionNode == RegionNode)
        {
            if (RangeInfo == Prev) /* Found at list head */
            {
                AcpiGbl_AddressRangeList[SpaceId] = RangeInfo->Next;
            }
            else
            {
                Prev->Next = RangeInfo->Next;
            }

            ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
                "\nRemoved [%4.4s] address range: 0x%p-0x%p\n",
                AcpiUtGetNodeName (RangeInfo->RegionNode),
                ACPI_CAST_PTR (void, RangeInfo->StartAddress),
                ACPI_CAST_PTR (void, RangeInfo->EndAddress)));

            ACPI_FREE (RangeInfo);
            return_VOID;
        }
```



```
        Prev = RangeInfo;
        RangeInfo = RangeInfo->Next;
    }

    return_VOID;
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtCheckAddressRange
 *
 * PARAMETERS:  SpaceId             - Address space ID
 *              Address             - Start address
 *              Length              - Length of address range
 *              Warn                - TRUE if warning on overlap desired
 *
 * RETURN:      Count of the number of conflicts detected. Zero is always
 *              returned for Space IDs other than Memory or I/O.
 *
 * DESCRIPTION: Check if the input address range overlaps any of the
 *              ASL operation region address ranges. The only supported
 *              Space IDs are Memory and I/O.
 *
 * MUTEX:       Assumes the namespace is locked.
 *
 ******************************************************************************/

UINT32
AcpiUtCheckAddressRange (
    ACPI_ADR_SPACE_TYPE     SpaceId,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  Length,
    BOOLEAN                 Warn)
{
    ACPI_ADDRESS_RANGE      *RangeInfo;
    ACPI_PHYSICAL_ADDRESS   EndAddress;
    char                    *Pathname;
    UINT32                  OverlapCount = 0;


    ACPI_FUNCTION_TRACE (UtCheckAddressRange);


    if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
        (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
    {
        return_UINT32 (0);
    }

    RangeInfo = AcpiGbl_AddressRangeList[SpaceId];
    EndAddress = Address + Length - 1;

    /* Check entire list for all possible conflicts */

    while (RangeInfo)
    {
        /*
         * Check if the requested address/length overlaps this
         * address range. There are four cases to consider:
         *
         * 1) Input address/length is contained completely in the
         *    address range
         * 2) Input address/length overlaps range at the range start
         * 3) Input address/length overlaps range at the range end
         * 4) Input address/length completely encompasses the range
```

```
259          */
260         if ((Address <= RangeInfo->EndAddress) &&
261             (EndAddress >= RangeInfo->StartAddress))
262         {
263             /* Found an address range overlap */

265             OverlapCount++;
266             if (Warn)   /* Optional warning message */
267             {
268                 Pathname = AcpiNsGetExternalPathname (RangeInfo->RegionNode);

270                 ACPI_WARNING ((AE_INFO,
271                     "%s range 0x%p-0x%p conflicts with OpRegion 0x%p-0x%p (%s)",
272                     AcpiUtGetRegionName (SpaceId),
273                     ACPI_CAST_PTR (void, Address),
274                     ACPI_CAST_PTR (void, EndAddress),
275                     ACPI_CAST_PTR (void, RangeInfo->StartAddress),
276                     ACPI_CAST_PTR (void, RangeInfo->EndAddress),
277                     Pathname));
278                 ACPI_FREE (Pathname);
279             }
280         }

282         RangeInfo = RangeInfo->Next;
283     }

285     return_UINT32 (OverlapCount);
286 }


289 /******************************************************************************
290  *
291  * FUNCTION:    AcpiUtDeleteAddressLists
292  *
293  * PARAMETERS:  None
294  *
295  * RETURN:      None
296  *
297  * DESCRIPTION: Delete all global address range lists (called during
298  *              subsystem shutdown).
299  *
300  *****************************************************************************/

302 void
303 AcpiUtDeleteAddressLists (
304     void)
305 {
306     ACPI_ADDRESS_RANGE      *Next;
307     ACPI_ADDRESS_RANGE      *RangeInfo;
308     int                     i;


311     /* Delete all elements in all address range lists */

313     for (i = 0; i < ACPI_ADDRESS_RANGE_MAX; i++)
314     {
315         Next = AcpiGbl_AddressRangeList[i];

317         while (Next)
318         {
319             RangeInfo = Next;
320             Next = RangeInfo->Next;
321             ACPI_FREE (RangeInfo);
322         }

324         AcpiGbl_AddressRangeList[i] = NULL;
```



```
325     }
326 }
```

```
**********************************************************
   10014 Thu Dec 26 13:49:35 2013
new/usr/src/common/acpica/components/utilities/utalloc.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utalloc - local memory allocation routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTALLOC_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acdebug.h"

  50 #define _COMPONENT          ACPI_UTILITIES
  51         ACPI_MODULE_NAME    ("utalloc")


  54 #if !defined (USE_NATIVE_ALLOCATE_ZEROED)
  55 /******************************************************************************
  56  *
  57  * FUNCTION:    AcpiOsAllocateZeroed
```



```
  58  *
  59  * PARAMETERS:  Size                - Size of the allocation
  60  *
  61  * RETURN:      Address of the allocated memory on success, NULL on failure.
  62  *
  63  * DESCRIPTION: Subsystem equivalent of calloc. Allocate and zero memory.
  64  *              This is the default implementation. Can be overridden via the
  65  *              USE_NATIVE_ALLOCATE_ZEROED flag.
  66  *
  67  *****************************************************************************/

  69 void *
  70 AcpiOsAllocateZeroed (
  71     ACPI_SIZE               Size)
  72 {
  73     void                    *Allocation;


  76     ACPI_FUNCTION_ENTRY ();


  79     Allocation = AcpiOsAllocate (Size);
  80     if (Allocation)
  81     {
  82         /* Clear the memory block */

  84         ACPI_MEMSET (Allocation, 0, Size);
  85     }

  87     return (Allocation);
  88 }

  90 #endif /* !USE_NATIVE_ALLOCATE_ZEROED */


  93 /******************************************************************************
  94  *
  95  * FUNCTION:    AcpiUtCreateCaches
  96  *
  97  * PARAMETERS:  None
  98  *
  99  * RETURN:      Status
 100  *
 101  * DESCRIPTION: Create all local caches
 102  *
 103  *****************************************************************************/

 105 ACPI_STATUS
 106 AcpiUtCreateCaches (
 107     void)
 108 {
 109     ACPI_STATUS             Status;


 112     /* Object Caches, for frequently used objects */

 114     Status = AcpiOsCreateCache ("Acpi-Namespace", sizeof (ACPI_NAMESPACE_NODE),
 115                 ACPI_MAX_NAMESPACE_CACHE_DEPTH, &AcpiGbl_NamespaceCache);
 116     if (ACPI_FAILURE (Status))
 117     {
 118         return (Status);
 119     }

 121     Status = AcpiOsCreateCache ("Acpi-State", sizeof (ACPI_GENERIC_STATE),
 122                 ACPI_MAX_STATE_CACHE_DEPTH, &AcpiGbl_StateCache);
 123     if (ACPI_FAILURE (Status))
```

```
124     {
125         return (Status);
126     }

128     Status = AcpiOsCreateCache ("Acpi-Parse", sizeof (ACPI_PARSE_OBJ_COMMON),
129                 ACPI_MAX_PARSE_CACHE_DEPTH, &AcpiGbl_PsNodeCache);
130     if (ACPI_FAILURE (Status))
131     {
132         return (Status);
133     }

135     Status = AcpiOsCreateCache ("Acpi-ParseExt", sizeof (ACPI_PARSE_OBJ_NAMED),
136                 ACPI_MAX_EXTPARSE_CACHE_DEPTH, &AcpiGbl_PsNodeExtCache);
137     if (ACPI_FAILURE (Status))
138     {
139         return (Status);
140     }

142     Status = AcpiOsCreateCache ("Acpi-Operand", sizeof (ACPI_OPERAND_OBJECT),
143                 ACPI_MAX_OBJECT_CACHE_DEPTH, &AcpiGbl_OperandCache);
144     if (ACPI_FAILURE (Status))
145     {
146         return (Status);
147     }


150 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

152     /* Memory allocation lists */

154     Status = AcpiUtCreateList ("Acpi-Global", 0,
155                 &AcpiGbl_GlobalList);
156     if (ACPI_FAILURE (Status))
157     {
158         return (Status);
159     }

161     Status = AcpiUtCreateList ("Acpi-Namespace", sizeof (ACPI_NAMESPACE_NODE),
162                 &AcpiGbl_NsNodeList);
163     if (ACPI_FAILURE (Status))
164     {
165         return (Status);
166     }
167 #endif

169     return (AE_OK);
170 }
```

*unchanged_portion_omitted_*

```
278 /******************************************************************************
279  *
280  * FUNCTION:    AcpiUtInitializeBuffer
281  *
282  * PARAMETERS:  Buffer              - Buffer to be validated
283  *              RequiredLength      - Length needed
284  *
285  * RETURN:      Status
286  *
287  * DESCRIPTION: Validate that the buffer is of the required length or
288  *              allocate a new buffer. Returned buffer is always zeroed.
289  *
290  *****************************************************************************/

292 ACPI_STATUS
293 AcpiUtInitializeBuffer (
```



```
294     ACPI_BUFFER             *Buffer,
295     ACPI_SIZE               RequiredLength)
296 {
297     ACPI_SIZE               InputBufferLength;


300     /* Parameter validation */

302     if (!Buffer || !RequiredLength)
303     {
304         return (AE_BAD_PARAMETER);
305     }

307     /*
308      * Buffer->Length is used as both an input and output parameter. Get the
309      * input actual length and set the output required buffer length.
310      */
311     InputBufferLength = Buffer->Length;
312     Buffer->Length = RequiredLength;

314     /*
315      * The input buffer length contains the actual buffer length, or the type
316      * of buffer to be allocated by this routine.
317      */
318     switch (InputBufferLength)
319     {
320     case ACPI_NO_BUFFER:

322         /* Return the exception (and the required buffer length) */

324         return (AE_BUFFER_OVERFLOW);

326     case ACPI_ALLOCATE_BUFFER:
327         /*
328          * Allocate a new buffer. We directectly call AcpiOsAllocate here to
329          * purposefully bypass the (optionally enabled) internal allocation
330          * tracking mechanism since we only want to track internal
331          * allocations. Note: The caller should use AcpiOsFree to free this
332          * buffer created via ACPI_ALLOCATE_BUFFER.
333          */

289         /* Allocate a new buffer */

334         Buffer->Pointer = AcpiOsAllocate (RequiredLength);
335         break;

337     case ACPI_ALLOCATE_LOCAL_BUFFER:

339         /* Allocate a new buffer with local interface to allow tracking */

341         Buffer->Pointer = ACPI_ALLOCATE (RequiredLength);
342         break;

344     default:

346         /* Existing buffer: Validate the size of the buffer */

348         if (InputBufferLength < RequiredLength)
349         {
350             return (AE_BUFFER_OVERFLOW);
351         }
352         break;
353     }

355     /* Validate allocation from above or input buffer pointer */
```

```
357     if (!Buffer->Pointer)
358     {
359         return (AE_NO_MEMORY);
360     }

362     /* Have a valid buffer, clear it */

364     ACPI_MEMSET (Buffer->Pointer, 0, RequiredLength);
365     return (AE_OK);
366 }


326 /*******************************************************************************
327  *
328  * FUNCTION:    AcpiUtAllocate
329  *
330  * PARAMETERS:  Size                - Size of the allocation
331  *              Component           - Component type of caller
332  *              Module              - Source file name of caller
333  *              Line                - Line number of caller
334  *
335  * RETURN:      Address of the allocated memory on success, NULL on failure.
336  *
337  * DESCRIPTION: Subsystem equivalent of malloc.
338  *
339  ******************************************************************************/

341 void *
342 AcpiUtAllocate (
343     ACPI_SIZE               Size,
344     UINT32                  Component,
345     const char              *Module,
346     UINT32                  Line)
347 {
348     void                    *Allocation;


351     ACPI_FUNCTION_TRACE_U32 (UtAllocate, Size);


354     /* Check for an inadvertent size of zero bytes */

356     if (!Size)
357     {
358         ACPI_WARNING ((Module, Line,
359             "Attempt to allocate zero bytes, allocating 1 byte"));
360         Size = 1;
361     }

363     Allocation = AcpiOsAllocate (Size);
364     if (!Allocation)
365     {
366         /* Report allocation error */

368         ACPI_WARNING ((Module, Line,
369             "Could not allocate size %u", (UINT32) Size));

371         return_PTR (NULL);
372     }

374     return_PTR (Allocation);
375 }


378 /*******************************************************************************
379  *
```



```
380  * FUNCTION:    AcpiUtAllocateZeroed
381  *
382  * PARAMETERS:  Size                - Size of the allocation
383  *              Component           - Component type of caller
384  *              Module              - Source file name of caller
385  *              Line                - Line number of caller
386  *
387  * RETURN:      Address of the allocated memory on success, NULL on failure.
388  *
389  * DESCRIPTION: Subsystem equivalent of calloc. Allocate and zero memory.
390  *
391  ******************************************************************************/

393 void *
394 AcpiUtAllocateZeroed (
395     ACPI_SIZE               Size,
396     UINT32                  Component,
397     const char              *Module,
398     UINT32                  Line)
399 {
400     void                    *Allocation;


403     ACPI_FUNCTION_ENTRY ();


406     Allocation = AcpiUtAllocate (Size, Component, Module, Line);
407     if (Allocation)
408     {
409         /* Clear the memory block */

411         ACPI_MEMSET (Allocation, 0, Size);
412     }

414     return (Allocation);
415 }
```

```
**********************************************************
    6838 Thu Dec 26 13:49:36 2013
new/usr/src/common/acpica/components/utilities/utbuffer.c
acpica-unix2-20130823
**********************************************************
/******************************************************************************
 *
 * Module Name: utbuffer - Buffer dump routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTBUFFER_C__

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utbuffer")


/******************************************************************************
 *
 * FUNCTION:    AcpiUtDumpBuffer
 *
 * PARAMETERS:  Buffer              - Buffer to dump
 *              Count               - Amount to dump, in bytes
 *              Display             - BYTE, WORD, DWORD, or QWORD display:
 *                                    DB_BYTE_DISPLAY
 *                                    DB_WORD_DISPLAY
```



```
 *                                    DB_DWORD_DISPLAY
 *                                    DB_QWORD_DISPLAY
 *              BaseOffset          - Beginning buffer offset (display only)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Generic dump buffer in both hex and ascii.
 *
 *****************************************************************************/

void
AcpiUtDumpBuffer (
    UINT8                   *Buffer,
    UINT32                  Count,
    UINT32                  Display,
    UINT32                  BaseOffset)
{
    UINT32                  i = 0;
    UINT32                  j;
    UINT32                  Temp32;
    UINT8                   BufChar;


    if (!Buffer)
    {
        AcpiOsPrintf ("Null Buffer Pointer in DumpBuffer!\n");
        return;
    }

    if ((Count < 4) || (Count & 0x01))
    {
        Display = DB_BYTE_DISPLAY;
    }

    /* Nasty little dump buffer routine! */

    while (i < Count)
    {
        /* Print current offset */

        AcpiOsPrintf ("%6.4X: ", (BaseOffset + i));

        /* Print 16 hex chars */

        for (j = 0; j < 16;)
        {
            if (i + j >= Count)
            {
                /* Dump fill spaces */

                AcpiOsPrintf ("%*s", ((Display * 2) + 1), " ");
                j += Display;
                continue;
            }

            switch (Display)
            {
            case DB_BYTE_DISPLAY:
            default:    /* Default is BYTE display */

                AcpiOsPrintf ("%02X ", Buffer[(ACPI_SIZE) i + j]);
                break;

            case DB_WORD_DISPLAY:

                ACPI_MOVE_16_TO_32 (&Temp32, &Buffer[(ACPI_SIZE) i + j]);
```

```
128                 AcpiOsPrintf ("%04X ", Temp32);
129                 break;

131             case DB_DWORD_DISPLAY:

133                 ACPI_MOVE_32_TO_32 (&Temp32, &Buffer[(ACPI_SIZE) i + j]);
134                 AcpiOsPrintf ("%08X ", Temp32);
135                 break;

137             case DB_QWORD_DISPLAY:

139                 ACPI_MOVE_32_TO_32 (&Temp32, &Buffer[(ACPI_SIZE) i + j]);
140                 AcpiOsPrintf ("%08X", Temp32);

142                 ACPI_MOVE_32_TO_32 (&Temp32, &Buffer[(ACPI_SIZE) i + j + 4]);
143                 AcpiOsPrintf ("%08X ", Temp32);
144                 break;
145             }

147             j += Display;
148         }

150         /*
151          * Print the ASCII equivalent characters but watch out for the bad
152          * unprintable ones (printable chars are 0x20 through 0x7E)
153          */
154         AcpiOsPrintf (" ");
155         for (j = 0; j < 16; j++)
156         {
157             if (i + j >= Count)
158             {
159                 AcpiOsPrintf ("\n");
160                 return;
161             }

163             BufChar = Buffer[(ACPI_SIZE) i + j];
164             if (ACPI_IS_PRINT (BufChar))
165             {
166                 AcpiOsPrintf ("%c", BufChar);
167             }
168             else
169             {
170                 AcpiOsPrintf (".");
171             }
172         }

174         /* Done with that line. */

176         AcpiOsPrintf ("\n");
177         i += 16;
178     }

180     return;
181 }


184 /*******************************************************************************
185  *
186  * FUNCTION:    AcpiUtDebugDumpBuffer
187  *
188  * PARAMETERS:  Buffer              - Buffer to dump
189  *              Count               - Amount to dump, in bytes
190  *              Display             - BYTE, WORD, DWORD, or QWORD display:
191  *                                    DB_BYTE_DISPLAY
192  *                                    DB_WORD_DISPLAY
193  *                                    DB_DWORD_DISPLAY
```



```
194  *                                    DB_QWORD_DISPLAY
195  *              ComponentID         - Caller's component ID
196  *
197  * RETURN:      None
198  *
199  * DESCRIPTION: Generic dump buffer in both hex and ascii.
200  *
201  ******************************************************************************/

203 void
204 AcpiUtDebugDumpBuffer (
205     UINT8                   *Buffer,
206     UINT32                  Count,
207     UINT32                  Display,
208     UINT32                  ComponentId)
209 {

211     /* Only dump the buffer if tracing is enabled */

213     if (!((ACPI_LV_TABLES & AcpiDbgLevel) &&
214         (ComponentId & AcpiDbgLayer)))
215     {
216         return;
217     }

219     AcpiUtDumpBuffer (Buffer, Count, Display, 0);
220 }
```

```
**********************************************************
    9247 Thu Dec 26 13:49:36 2013
new/usr/src/common/acpica/components/utilities/utcache.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utcache - local cache allocation routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTCACHE_C__

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utcache")


#ifdef ACPI_USE_LOCAL_CACHE
/******************************************************************************
 *
 * FUNCTION:    AcpiOsCreateCache
 *
 * PARAMETERS:  CacheName       - Ascii name for the cache
```



```
 *              ObjectSize      - Size of each cached object
 *              MaxDepth        - Maximum depth of the cache (in objects)
 *              ReturnCache     - Where the new cache object is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create a cache object
 *
 ******************************************************************************/

ACPI_STATUS
AcpiOsCreateCache (
    char                    *CacheName,
    UINT16                  ObjectSize,
    UINT16                  MaxDepth,
    ACPI_MEMORY_LIST        **ReturnCache)
{
    ACPI_MEMORY_LIST        *Cache;


    ACPI_FUNCTION_ENTRY ();


    if (!CacheName || !ReturnCache || (ObjectSize < 16))
    {
        return (AE_BAD_PARAMETER);
    }

    /* Create the cache object */

    Cache = AcpiOsAllocate (sizeof (ACPI_MEMORY_LIST));
    if (!Cache)
    {
        return (AE_NO_MEMORY);
    }

    /* Populate the cache object and return it */

    ACPI_MEMSET (Cache, 0, sizeof (ACPI_MEMORY_LIST));
    Cache->LinkOffset = 8;
    Cache->ListName   = CacheName;
    Cache->ObjectSize = ObjectSize;
    Cache->MaxDepth   = MaxDepth;

    *ReturnCache = Cache;
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsPurgeCache
 *
 * PARAMETERS:  Cache           - Handle to cache object
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Free all objects within the requested cache.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiOsPurgeCache (
    ACPI_MEMORY_LIST        *Cache)
{
    void                    *Next;
```

```
124     char                    *Next;
124     ACPI_STATUS             Status;


127     ACPI_FUNCTION_ENTRY ();


130     if (!Cache)
131     {
132         return (AE_BAD_PARAMETER);
133     }

135     Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
136     if (ACPI_FAILURE (Status))
137     {
138         return (Status);
139     }

141     /* Walk the list of objects in this cache */

143     while (Cache->ListHead)
144     {
145         /* Delete and unlink one cached state object */

147         Next = ACPI_GET_DESCRIPTOR_PTR (Cache->ListHead);
148         Next = *(ACPI_CAST_INDIRECT_PTR (char,
149                     &(((char *) Cache->ListHead)[Cache->LinkOffset])));
148         ACPI_FREE (Cache->ListHead);

150         Cache->ListHead = Next;
151         Cache->CurrentDepth--;
152     }

154     (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
155     return (AE_OK);
156 }
_____unchanged_portion_omitted_


197 /******************************************************************************
198  *
199  * FUNCTION:    AcpiOsReleaseObject
200  *
201  * PARAMETERS:  Cache       - Handle to cache object
202  *              Object      - The object to be released
203  *
204  * RETURN:      None
205  *
206  * DESCRIPTION: Release an object to the specified cache. If cache is full,
207  *              the object is deleted.
208  *
209  *****************************************************************************/

211 ACPI_STATUS
212 AcpiOsReleaseObject (
213     ACPI_MEMORY_LIST        *Cache,
214     void                    *Object)
215 {
216     ACPI_STATUS             Status;


219     ACPI_FUNCTION_ENTRY ();


222     if (!Cache || !Object)
223     {
```



```
224         return (AE_BAD_PARAMETER);
225     }

227     /* If cache is full, just free this object */

229     if (Cache->CurrentDepth >= Cache->MaxDepth)
230     {
231         ACPI_FREE (Object);
232         ACPI_MEM_TRACKING (Cache->TotalFreed++);
233     }

235     /* Otherwise put this object back into the cache */

237     else
238     {
239         Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
240         if (ACPI_FAILURE (Status))
241         {
242             return (Status);
243         }

245         /* Mark the object as cached */

247         ACPI_MEMSET (Object, 0xCA, Cache->ObjectSize);
248         ACPI_SET_DESCRIPTOR_TYPE (Object, ACPI_DESC_TYPE_CACHED);

250         /* Put the object at the head of the cache list */

252         ACPI_SET_DESCRIPTOR_PTR (Object, Cache->ListHead);
254         * (ACPI_CAST_INDIRECT_PTR (char,
255             &(((char *) Object)[Cache->LinkOffset]))) = Cache->ListHead;
253         Cache->ListHead = Object;
254         Cache->CurrentDepth++;

256         (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
257     }

259     return (AE_OK);
260 }


263 /******************************************************************************
264  *
265  * FUNCTION:    AcpiOsAcquireObject
266  *
267  * PARAMETERS:  Cache           - Handle to cache object
268  *
269  * RETURN:      the acquired object. NULL on error
270  *
271  * DESCRIPTION: Get an object from the specified cache. If cache is empty,
272  *              the object is allocated.
273  *
274  *****************************************************************************/

276 void *
277 AcpiOsAcquireObject (
278     ACPI_MEMORY_LIST        *Cache)
279 {
280     ACPI_STATUS             Status;
281     void                    *Object;


284     ACPI_FUNCTION_NAME (OsAcquireObject);


287     if (!Cache)
```

```
288     {
289         return_PTR (NULL);
292         return (NULL);
290     }

292     Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
293     if (ACPI_FAILURE (Status))
294     {
295         return_PTR (NULL);
298         return (NULL);
296     }

298     ACPI_MEM_TRACKING (Cache->Requests++);

300     /* Check the cache first */

302     if (Cache->ListHead)
303     {
304         /* There is an object available, use it */

306         Object = Cache->ListHead;
307         Cache->ListHead = ACPI_GET_DESCRIPTOR_PTR (Object);
310         Cache->ListHead = *(ACPI_CAST_INDIRECT_PTR (char,
311                                 &(((char *) Object)[Cache->LinkOffset])));

309         Cache->CurrentDepth--;

311         ACPI_MEM_TRACKING (Cache->Hits++);
312         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
313             "Object %p from %s cache\n", Object, Cache->ListName));

315         Status = AcpiUtReleaseMutex (ACPI_MTX_CACHES);
316         if (ACPI_FAILURE (Status))
317         {
318             return_PTR (NULL);
322             return (NULL);
319         }

321         /* Clear (zero) the previously used Object */

323         ACPI_MEMSET (Object, 0, Cache->ObjectSize);
324     }
325     else
326     {
327         /* The cache is empty, create a new object */

329         ACPI_MEM_TRACKING (Cache->TotalAllocated++);

331 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
332         if ((Cache->TotalAllocated - Cache->TotalFreed) > Cache->MaxOccupied)
333         {
334             Cache->MaxOccupied = Cache->TotalAllocated - Cache->TotalFreed;
335         }
336 #endif

338         /* Avoid deadlock with ACPI_ALLOCATE_ZEROED */

340         Status = AcpiUtReleaseMutex (ACPI_MTX_CACHES);
341         if (ACPI_FAILURE (Status))
342         {
343             return_PTR (NULL);
347             return (NULL);
344         }

346         Object = ACPI_ALLOCATE_ZEROED (Cache->ObjectSize);
347         if (!Object)
```



```
348         {
349             return_PTR (NULL);
353             return (NULL);
350         }
351     }

353     return_PTR (Object);
357     return (Object);
354 }
355 #endif /* ACPI_USE_LOCAL_CACHE */
```

```
**********************************************************
   24537 Thu Dec 26 13:49:36 2013
new/usr/src/common/acpica/components/utilities/utclib.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: cmclib - Local implementation of C library functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __CMCLIB_C__

#include "acpi.h"
#include "accommon.h"

/*
 * These implementations of standard C Library routines can optionally be
 * used if a C library is not available. In general, they are less efficient
 * than an inline or assembly implementation
 */

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("cmclib")


#ifndef ACPI_USE_SYSTEM_CLIBRARY
```



```
#define NEGATIVE    1
#define POSITIVE    0


/******************************************************************************
 *
 * FUNCTION:    AcpiUtMemcmp (memcmp)
 *
 * PARAMETERS:  Buffer1         - First Buffer
 *              Buffer2         - Second Buffer
 *              Count           - Maximum # of bytes to compare
 *
 * RETURN:      Index where Buffers mismatched, or 0 if Buffers matched
 *
 * DESCRIPTION: Compare two Buffers, with a maximum length
 *
 *****************************************************************************/

int
AcpiUtMemcmp (
    const char              *Buffer1,
    const char              *Buffer2,
    ACPI_SIZE               Count)
{

    for ( ; Count-- && (*Buffer1 == *Buffer2); Buffer1++, Buffer2++)
    {
    }

    return ((Count == ACPI_SIZE_MAX) ? 0 : ((unsigned char) *Buffer1 -
        (unsigned char) *Buffer2));
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtMemcpy (memcpy)
 *
 * PARAMETERS:  Dest        - Target of the copy
 *              Src         - Source buffer to copy
 *              Count       - Number of bytes to copy
 *
 * RETURN:      Dest
 *
 * DESCRIPTION: Copy arbitrary bytes of memory
 *
 *****************************************************************************/

void *
AcpiUtMemcpy (
    void                    *Dest,
    const void              *Src,
    ACPI_SIZE               Count)
{
    char                    *New = (char *) Dest;
    char                    *Old = (char *) Src;


    while (Count)
    {
        *New = *Old;
        New++;
        Old++;
        Count--;
    }
```

```
128     return (Dest);
129 }


132 /******************************************************************************
133  *
134  * FUNCTION:    AcpiUtMemset (memset)
135  *
136  * PARAMETERS:  Dest        - Buffer to set
137  *              Value       - Value to set each byte of memory
138  *              Count       - Number of bytes to set
139  *
140  * RETURN:      Dest
141  *
142  * DESCRIPTION: Initialize a buffer to a known value.
143  *
144  *****************************************************************************/

146 void *
147 AcpiUtMemset (
148     void                    *Dest,
149     UINT8                   Value,
150     ACPI_SIZE               Count)
151 {
152     char                    *New = (char *) Dest;


155     while (Count)
156     {
157         *New = (char) Value;
158         New++;
159         Count--;
160     }

162     return (Dest);
163 }


166 /******************************************************************************
167  *
168  * FUNCTION:    AcpiUtStrlen (strlen)
169  *
170  * PARAMETERS:  String              - Null terminated string
171  *
172  * RETURN:      Length
173  *
174  * DESCRIPTION: Returns the length of the input string
175  *
176  *****************************************************************************/


179 ACPI_SIZE
180 AcpiUtStrlen (
181     const char              *String)
182 {
183     UINT32                  Length = 0;


186     /* Count the string until a null is encountered */

188     while (*String)
189     {
190         Length++;
191         String++;
192     }
```



```
194     return (Length);
195 }


198 /******************************************************************************
199  *
200  * FUNCTION:    AcpiUtStrcpy (strcpy)
201  *
202  * PARAMETERS:  DstString       - Target of the copy
203  *              SrcString       - The source string to copy
204  *
205  * RETURN:      DstString
206  *
207  * DESCRIPTION: Copy a null terminated string
208  *
209  *****************************************************************************/

211 char *
212 AcpiUtStrcpy (
213     char                    *DstString,
214     const char              *SrcString)
215 {
216     char                    *String = DstString;


219     /* Move bytes brute force */

221     while (*SrcString)
222     {
223         *String = *SrcString;

225         String++;
226         SrcString++;
227     }

229     /* Null terminate */

231     *String = 0;
232     return (DstString);
233 }


236 /******************************************************************************
237  *
238  * FUNCTION:    AcpiUtStrncpy (strncpy)
239  *
240  * PARAMETERS:  DstString       - Target of the copy
241  *              SrcString       - The source string to copy
242  *              Count           - Maximum # of bytes to copy
243  *
244  * RETURN:      DstString
245  *
246  * DESCRIPTION: Copy a null terminated string, with a maximum length
247  *
248  *****************************************************************************/

250 char *
251 AcpiUtStrncpy (
252     char                    *DstString,
253     const char              *SrcString,
254     ACPI_SIZE               Count)
255 {
256     char                    *String = DstString;
```

```
    /* Copy the string */

    for (String = DstString;
         Count && (Count--, (*String++ = *SrcString++)); )
    {;}

    /* Pad with nulls if necessary */

    while (Count--)
    {
        *String = 0;
        String++;
    }

    /* Return original pointer */

    return (DstString);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrcmp (strcmp)
 *
 * PARAMETERS:  String1         - First string
 *              String2         - Second string
 *
 * RETURN:      Index where strings mismatched, or 0 if strings matched
 *
 * DESCRIPTION: Compare two null terminated strings
 *
 ******************************************************************************/

int
AcpiUtStrcmp (
    const char              *String1,
    const char              *String2)
{


    for ( ; (*String1 == *String2); String2++)
    {
        if (!*String1++)
        {
            return (0);
        }
    }

    return ((unsigned char) *String1 - (unsigned char) *String2);
}


#ifdef ACPI_FUTURE_IMPLEMENTATION
/* Not used at this time */
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrchr (strchr)
 *
 * PARAMETERS:  String          - Search string
 *              ch              - character to search for
 *
 * RETURN:      Ptr to char or NULL if not found
 *
 * DESCRIPTION: Search a string for a character
 *
 ******************************************************************************/
```



```
char *
AcpiUtStrchr (
    const char              *String,
    int                     ch)
{


    for ( ; (*String); String++)
    {
        if ((*String) == (char) ch)
        {
            return ((char *) String);
        }
    }

    return (NULL);
}
#endif

/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrncmp (strncmp)
 *
 * PARAMETERS:  String1         - First string
 *              String2         - Second string
 *              Count           - Maximum # of bytes to compare
 *
 * RETURN:      Index where strings mismatched, or 0 if strings matched
 *
 * DESCRIPTION: Compare two null terminated strings, with a maximum length
 *
 ******************************************************************************/

int
AcpiUtStrncmp (
    const char              *String1,
    const char              *String2,
    ACPI_SIZE               Count)
{


    for ( ; Count-- && (*String1 == *String2); String2++)
    {
        if (!*String1++)
        {
            return (0);
        }
    }

    return ((Count == ACPI_SIZE_MAX) ? 0 : ((unsigned char) *String1 -
        (unsigned char) *String2));
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrcat (Strcat)
 *
 * PARAMETERS:  DstString       - Target of the copy
 *              SrcString       - The source string to copy
 *
 * RETURN:      DstString
 *
 * DESCRIPTION: Append a null terminated string to a null terminated string
 *
```

```
 *****************************************************************************/

char *
AcpiUtStrcat (
    char                    *DstString,
    const char              *SrcString)
{
    char                    *String;


    /* Find end of the destination string */

    for (String = DstString; *String++; )
    { ; }

    /* Concatenate the string */

    for (--String; (*String++ = *SrcString++); )
    { ; }

    return (DstString);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiUtStrncat (strncat)
 *
 * PARAMETERS:  DstString       - Target of the copy
 *              SrcString       - The source string to copy
 *              Count           - Maximum # of bytes to copy
 *
 * RETURN:      DstString
 *
 * DESCRIPTION: Append a null terminated string to a null terminated string,
 *              with a maximum count.
 *
 *****************************************************************************/

char *
AcpiUtStrncat (
    char                    *DstString,
    const char              *SrcString,
    ACPI_SIZE               Count)
{
    char                    *String;


    if (Count)
    {
        /* Find end of the destination string */

        for (String = DstString; *String++; )
        { ; }

        /* Concatenate the string */

        for (--String; (*String++ = *SrcString++) && --Count; )
        { ; }

        /* Null terminate if necessary */

        if (!Count)
        {
            *String = 0;
        }
```



```
    }

    return (DstString);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiUtStrstr (strstr)
 *
 * PARAMETERS:  String1         - Target string
 *              String2         - Substring to search for
 *
 * RETURN:      Where substring match starts, Null if no match found
 *
 * DESCRIPTION: Checks if String2 occurs in String1. This is not really a
 *              full implementation of strstr, only sufficient for command
 *              matching
 *
 *****************************************************************************/

char *
AcpiUtStrstr (
    char                    *String1,
    char                    *String2)
{
    char                    *String;


    if (AcpiUtStrlen (String2) > AcpiUtStrlen (String1))
    {
        return (NULL);
    }

    /* Walk entire string, comparing the letters */

    for (String = String1; *String2; )
    {
        if (*String2 != *String)
        {
            return (NULL);
        }

        String2++;
        String++;
    }

    return (String1);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiUtStrtoul (strtoul)
 *
 * PARAMETERS:  String          - Null terminated string
 *              Terminater      - Where a pointer to the terminating byte is
 *                                returned
 *              Base            - Radix of the string
 *
 * RETURN:      Converted value
 *
 * DESCRIPTION: Convert a string into a 32-bit unsigned value.
 *              Note: use AcpiUtStrtoul64 for 64-bit integers.
 *
 *****************************************************************************/
```

```
524 UINT32
525 AcpiUtStrtoul (
526     const char              *String,
527     char                    **Terminator,
528     UINT32                  Base)
529 {
530     UINT32                  converted = 0;
531     UINT32                  index;
532     UINT32                  sign;
533     const char              *StringStart;
534     UINT32                  ReturnValue = 0;
535     ACPI_STATUS             Status = AE_OK;


538     /*
539      * Save the value of the pointer to the buffer's first
540      * character, save the current errno value, and then
541      * skip over any white space in the buffer:
542      */
543     StringStart = String;
544     while (ACPI_IS_SPACE (*String) || *String == '\t')
545     {
546         ++String;
547     }

549     /*
550      * The buffer may contain an optional plus or minus sign.
551      * If it does, then skip over it but remember what is was:
552      */
553     if (*String == '-')
554     {
555         sign = NEGATIVE;
556         ++String;
557     }
558     else if (*String == '+')
559     {
560         ++String;
561         sign = POSITIVE;
562     }
563     else
564     {
565         sign = POSITIVE;
566     }

568     /*
569      * If the input parameter Base is zero, then we need to
570      * determine if it is octal, decimal, or hexadecimal:
571      */
572     if (Base == 0)
573     {
574         if (*String == '0')
575         {
576             if (AcpiUtToLower (*(++String)) == 'x')
577             {
578                 Base = 16;
579                 ++String;
580             }
581             else
582             {
583                 Base = 8;
584             }
585         }
586         else
587         {
588             Base = 10;
```



```
589         }
590     }
591     else if (Base < 2 || Base > 36)
592     {
593         /*
594          * The specified Base parameter is not in the domain of
595          * this function:
596          */
597         goto done;
598     }

600     /*
601      * For octal and hexadecimal bases, skip over the leading
602      * 0 or 0x, if they are present.
603      */
604     if (Base == 8 && *String == '0')
605     {
606         String++;
607     }

609     if (Base == 16 &&
610         *String == '0' &&
611         AcpiUtToLower (*(++String)) == 'x')
612     {
613         String++;
614     }

616     /*
617      * Main loop: convert the string to an unsigned long:
618      */
619     while (*String)
620     {
621         if (ACPI_IS_DIGIT (*String))
622         {
623             index = (UINT32) ((UINT8) *String - '0');
624         }
625         else
626         {
627             index = (UINT32) AcpiUtToUpper (*String);
628             if (ACPI_IS_UPPER (index))
629             {
630                 index = index - 'A' + 10;
631             }
632             else
633             {
634                 goto done;
635             }
636         }

638         if (index >= Base)
639         {
640             goto done;
641         }

643         /*
644          * Check to see if value is out of range:
645          */

647         if (ReturnValue > ((ACPI_UINT32_MAX - (UINT32) index) /
648                            (UINT32) Base))
649         {
650             Status = AE_ERROR;
651             ReturnValue = 0;           /* reset */
652         }
653         else
654         {
```

```
655             ReturnValue *= Base;
656             ReturnValue += index;
657             converted = 1;
658         }

660         ++String;
661     }

663 done:
664     /*
665      * If appropriate, update the caller's pointer to the next
666      * unconverted character in the buffer.
667      */
668     if (Terminator)
669     {
670         if (converted == 0 && ReturnValue == 0 && String != NULL)
671         {
672             *Terminator = (char *) StringStart;
673         }
674         else
675         {
676             *Terminator = (char *) String;
677         }
678     }

680     if (Status == AE_ERROR)
681     {
682         ReturnValue = ACPI_UINT32_MAX;
683     }

685     /*
686      * If a minus sign was present, then "the conversion is negated":
687      */
688     if (sign == NEGATIVE)
689     {
690         ReturnValue = (ACPI_UINT32_MAX - ReturnValue) + 1;
691     }

693     return (ReturnValue);
694 }


697 /*******************************************************************************
698  *
699  * FUNCTION:    AcpiUtToUpper (TOUPPER)
700  *
701  * PARAMETERS:  c           - Character to convert
702  *
703  * RETURN:      Converted character as an int
704  *
705  * DESCRIPTION: Convert character to uppercase
706  *
707  ******************************************************************************/

709 int
710 AcpiUtToUpper (
711     int                 c)
712 {

714     return (ACPI_IS_LOWER(c) ? ((c)-0x20) : (c));
715 }


718 /*******************************************************************************
719  *
720  * FUNCTION:    AcpiUtToLower (TOLOWER)
```



```
721  *
722  * PARAMETERS:  c           - Character to convert
723  *
724  * RETURN:      Converted character as an int
725  *
726  * DESCRIPTION: Convert character to lowercase
727  *
728  ******************************************************************************/

730 int
731 AcpiUtToLower (
732     int                 c)
733 {

735     return (ACPI_IS_UPPER(c) ? ((c)+0x20) : (c));
736 }


739 /*******************************************************************************
740  *
741  * FUNCTION:    is* functions
742  *
743  * DESCRIPTION: is* functions use the ctype table below
744  *
745  ******************************************************************************/

747 const UINT8 _acpi_ctype[257] = {
748     _ACPI_CN,               /* 0x00     0 NUL */
749     _ACPI_CN,               /* 0x01     1 SOH */
750     _ACPI_CN,               /* 0x02     2 STX */
751     _ACPI_CN,               /* 0x03     3 ETX */
752     _ACPI_CN,               /* 0x04     4 EOT */
753     _ACPI_CN,               /* 0x05     5 ENQ */
754     _ACPI_CN,               /* 0x06     6 ACK */
755     _ACPI_CN,               /* 0x07     7 BEL */
756     _ACPI_CN,               /* 0x08     8 BS  */
757     _ACPI_CN|_ACPI_SP,      /* 0x09     9 TAB */
758     _ACPI_CN|_ACPI_SP,      /* 0x0A    10 LF  */
759     _ACPI_CN|_ACPI_SP,      /* 0x0B    11 VT  */
760     _ACPI_CN|_ACPI_SP,      /* 0x0C    12 FF  */
761     _ACPI_CN|_ACPI_SP,      /* 0x0D    13 CR  */
762     _ACPI_CN,               /* 0x0E    14 SO  */
763     _ACPI_CN,               /* 0x0F    15 SI  */
764     _ACPI_CN,               /* 0x10    16 DLE */
765     _ACPI_CN,               /* 0x11    17 DC1 */
766     _ACPI_CN,               /* 0x12    18 DC2 */
767     _ACPI_CN,               /* 0x13    19 DC3 */
768     _ACPI_CN,               /* 0x14    20 DC4 */
769     _ACPI_CN,               /* 0x15    21 NAK */
770     _ACPI_CN,               /* 0x16    22 SYN */
771     _ACPI_CN,               /* 0x17    23 ETB */
772     _ACPI_CN,               /* 0x18    24 CAN */
773     _ACPI_CN,               /* 0x19    25 EM  */
774     _ACPI_CN,               /* 0x1A    26 SUB */
775     _ACPI_CN,               /* 0x1B    27 ESC */
776     _ACPI_CN,               /* 0x1C    28 FS  */
777     _ACPI_CN,               /* 0x1D    29 GS  */
778     _ACPI_CN,               /* 0x1E    30 RS  */
779     _ACPI_CN,               /* 0x1F    31 US  */
780     _ACPI_XS|_ACPI_SP,      /* 0x20    32 ' ' */
781     _ACPI_PU,               /* 0x21    33 '!' */
782     _ACPI_PU,               /* 0x22    34 '"' */
783     _ACPI_PU,               /* 0x23    35 '#' */
784     _ACPI_PU,               /* 0x24    36 '$' */
785     _ACPI_PU,               /* 0x25    37 '%' */
786     _ACPI_PU,               /* 0x26    38 '&' */
```

```
    _ACPI_PU,           /* 0x27    39 ''' */
    _ACPI_PU,           /* 0x28    40 '(' */
    _ACPI_PU,           /* 0x29    41 ')' */
    _ACPI_PU,           /* 0x2A    42 '*' */
    _ACPI_PU,           /* 0x2B    43 '+' */
    _ACPI_PU,           /* 0x2C    44 ',' */
    _ACPI_PU,           /* 0x2D    45 '-' */
    _ACPI_PU,           /* 0x2E    46 '.' */
    _ACPI_PU,           /* 0x2F    47 '/' */
    _ACPI_XD|_ACPI_DI,  /* 0x30    48 '0' */
    _ACPI_XD|_ACPI_DI,  /* 0x31    49 '1' */
    _ACPI_XD|_ACPI_DI,  /* 0x32    50 '2' */
    _ACPI_XD|_ACPI_DI,  /* 0x33    51 '3' */
    _ACPI_XD|_ACPI_DI,  /* 0x34    52 '4' */
    _ACPI_XD|_ACPI_DI,  /* 0x35    53 '5' */
    _ACPI_XD|_ACPI_DI,  /* 0x36    54 '6' */
    _ACPI_XD|_ACPI_DI,  /* 0x37    55 '7' */
    _ACPI_XD|_ACPI_DI,  /* 0x38    56 '8' */
    _ACPI_XD|_ACPI_DI,  /* 0x39    57 '9' */
    _ACPI_PU,           /* 0x3A    58 ':' */
    _ACPI_PU,           /* 0x3B    59 ';' */
    _ACPI_PU,           /* 0x3C    60 '<' */
    _ACPI_PU,           /* 0x3D    61 '=' */
    _ACPI_PU,           /* 0x3E    62 '>' */
    _ACPI_PU,           /* 0x3F    63 '?' */
    _ACPI_PU,           /* 0x40    64 '@' */
    _ACPI_XD|_ACPI_UP,  /* 0x41    65 'A' */
    _ACPI_XD|_ACPI_UP,  /* 0x42    66 'B' */
    _ACPI_XD|_ACPI_UP,  /* 0x43    67 'C' */
    _ACPI_XD|_ACPI_UP,  /* 0x44    68 'D' */
    _ACPI_XD|_ACPI_UP,  /* 0x45    69 'E' */
    _ACPI_XD|_ACPI_UP,  /* 0x46    70 'F' */
    _ACPI_UP,           /* 0x47    71 'G' */
    _ACPI_UP,           /* 0x48    72 'H' */
    _ACPI_UP,           /* 0x49    73 'I' */
    _ACPI_UP,           /* 0x4A    74 'J' */
    _ACPI_UP,           /* 0x4B    75 'K' */
    _ACPI_UP,           /* 0x4C    76 'L' */
    _ACPI_UP,           /* 0x4D    77 'M' */
    _ACPI_UP,           /* 0x4E    78 'N' */
    _ACPI_UP,           /* 0x4F    79 'O' */
    _ACPI_UP,           /* 0x50    80 'P' */
    _ACPI_UP,           /* 0x51    81 'Q' */
    _ACPI_UP,           /* 0x52    82 'R' */
    _ACPI_UP,           /* 0x53    83 'S' */
    _ACPI_UP,           /* 0x54    84 'T' */
    _ACPI_UP,           /* 0x55    85 'U' */
    _ACPI_UP,           /* 0x56    86 'V' */
    _ACPI_UP,           /* 0x57    87 'W' */
    _ACPI_UP,           /* 0x58    88 'X' */
    _ACPI_UP,           /* 0x59    89 'Y' */
    _ACPI_UP,           /* 0x5A    90 'Z' */
    _ACPI_PU,           /* 0x5B    91 '[' */
    _ACPI_PU,           /* 0x5C    92 '\' */
    _ACPI_PU,           /* 0x5D    93 ']' */
    _ACPI_PU,           /* 0x5E    94 '^' */
    _ACPI_PU,           /* 0x5F    95 '_' */
    _ACPI_PU,           /* 0x60    96 '`' */
    _ACPI_XD|_ACPI_LO,  /* 0x61    97 'a' */
    _ACPI_XD|_ACPI_LO,  /* 0x62    98 'b' */
    _ACPI_XD|_ACPI_LO,  /* 0x63    99 'c' */
    _ACPI_XD|_ACPI_LO,  /* 0x64   100 'd' */
    _ACPI_XD|_ACPI_LO,  /* 0x65   101 'e' */
    _ACPI_XD|_ACPI_LO,  /* 0x66   102 'f' */
    _ACPI_LO,           /* 0x67   103 'g' */
    _ACPI_LO,           /* 0x68   104 'h' */
```



```
    _ACPI_LO,           /* 0x69   105 'i' */
    _ACPI_LO,           /* 0x6A   106 'j' */
    _ACPI_LO,           /* 0x6B   107 'k' */
    _ACPI_LO,           /* 0x6C   108 'l' */
    _ACPI_LO,           /* 0x6D   109 'm' */
    _ACPI_LO,           /* 0x6E   110 'n' */
    _ACPI_LO,           /* 0x6F   111 'o' */
    _ACPI_LO,           /* 0x70   112 'p' */
    _ACPI_LO,           /* 0x71   113 'q' */
    _ACPI_LO,           /* 0x72   114 'r' */
    _ACPI_LO,           /* 0x73   115 's' */
    _ACPI_LO,           /* 0x74   116 't' */
    _ACPI_LO,           /* 0x75   117 'u' */
    _ACPI_LO,           /* 0x76   118 'v' */
    _ACPI_LO,           /* 0x77   119 'w' */
    _ACPI_LO,           /* 0x78   120 'x' */
    _ACPI_LO,           /* 0x79   121 'y' */
    _ACPI_LO,           /* 0x7A   122 'z' */
    _ACPI_PU,           /* 0x7B   123 '{' */
    _ACPI_PU,           /* 0x7C   124 '|' */
    _ACPI_PU,           /* 0x7D   125 '}' */
    _ACPI_PU,           /* 0x7E   126 '~' */
    _ACPI_CN,           /* 0x7F   127 DEL */

    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0x80 to 0x8F    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0x90 to 0x9F    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xA0 to 0xAF    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xB0 to 0xBF    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xC0 to 0xCF    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xD0 to 0xDF    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xE0 to 0xEF    */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  /* 0xF0 to 0xFF    */
    0                                 /* 0x100 */
};


#endif /* ACPI_USE_SYSTEM_CLIBRARY */
```

```
**********************************************************
   31384 Thu Dec 26 13:49:37 2013
new/usr/src/common/acpica/components/utilities/utcopy.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utcopy - Internal to external object translation utilities
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTCOPY_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acnamesp.h"


  51 #define _COMPONENT          ACPI_UTILITIES
  52         ACPI_MODULE_NAME    ("utcopy")

  54 /* Local prototypes */

  56 static ACPI_STATUS
  57 AcpiUtCopyIsimpleToEsimple (
  58     ACPI_OPERAND_OBJECT     *InternalObject,
  59     ACPI_OBJECT             *ExternalObject,
```



```
  60     UINT8                   *DataSpace,
  61     ACPI_SIZE               *BufferSpaceUsed);

  63 static ACPI_STATUS
  64 AcpiUtCopyIelementToIelement (
  65     UINT8                   ObjectType,
  66     ACPI_OPERAND_OBJECT     *SourceObject,
  67     ACPI_GENERIC_STATE      *State,
  68     void                    *Context);

  70 static ACPI_STATUS
  71 AcpiUtCopyIpackageToEpackage (
  72     ACPI_OPERAND_OBJECT     *InternalObject,
  73     UINT8                   *Buffer,
  74     ACPI_SIZE               *SpaceUsed);

  76 static ACPI_STATUS
  77 AcpiUtCopyEsimpleToIsimple(
  78     ACPI_OBJECT             *UserObj,
  79     ACPI_OPERAND_OBJECT     **ReturnObj);

  81 static ACPI_STATUS
  82 AcpiUtCopyEpackageToIpackage (
  83     ACPI_OBJECT             *ExternalObject,
  84     ACPI_OPERAND_OBJECT     **InternalObject);

  86 static ACPI_STATUS
  87 AcpiUtCopySimpleObject (
  88     ACPI_OPERAND_OBJECT     *SourceDesc,
  89     ACPI_OPERAND_OBJECT     *DestDesc);

  91 static ACPI_STATUS
  92 AcpiUtCopyIelementToEelement (
  93     UINT8                   ObjectType,
  94     ACPI_OPERAND_OBJECT     *SourceObject,
  95     ACPI_GENERIC_STATE      *State,
  96     void                    *Context);

  98 static ACPI_STATUS
  99 AcpiUtCopyIpackageToIpackage (
 100     ACPI_OPERAND_OBJECT     *SourceObj,
 101     ACPI_OPERAND_OBJECT     *DestObj,
 102     ACPI_WALK_STATE         *WalkState);


 105 /*******************************************************************************
 106  *
 107  * FUNCTION:    AcpiUtCopyIsimpleToEsimple
 108  *
 109  * PARAMETERS:  InternalObject      - Source object to be copied
 110  *              ExternalObject      - Where to return the copied object
 111  *              DataSpace           - Where object data is returned (such as
 112  *                                    buffer and string data)
 113  *              BufferSpaceUsed     - Length of DataSpace that was used
 114  *
 115  * RETURN:      Status
 116  *
 117  * DESCRIPTION: This function is called to copy a simple internal object to
 118  *              an external object.
 119  *
 120  *              The DataSpace buffer is assumed to have sufficient space for
 121  *              the object.
 122  *
 123  ******************************************************************************/

 125 static ACPI_STATUS
```

```
126 AcpiUtCopyIsimpleToEsimple (
127     ACPI_OPERAND_OBJECT     *InternalObject,
128     ACPI_OBJECT             *ExternalObject,
129     UINT8                   *DataSpace,
130     ACPI_SIZE               *BufferSpaceUsed)
131 {
132     ACPI_STATUS             Status = AE_OK;


135     ACPI_FUNCTION_TRACE (UtCopyIsimpleToEsimple);


138     *BufferSpaceUsed = 0;

140     /*
141      * Check for NULL object case (could be an uninitialized
142      * package element)
143      */
144     if (!InternalObject)
145     {
146         return_ACPI_STATUS (AE_OK);
147     }

149     /* Always clear the external object */

151     ACPI_MEMSET (ExternalObject, 0, sizeof (ACPI_OBJECT));

153     /*
154      * In general, the external object will be the same type as
155      * the internal object
156      */
157     ExternalObject->Type = InternalObject->Common.Type;

159     /* However, only a limited number of external types are supported */

161     switch (InternalObject->Common.Type)
162     {
163     case ACPI_TYPE_STRING:

165         ExternalObject->String.Pointer = (char *) DataSpace;
166         ExternalObject->String.Length  = InternalObject->String.Length;
167         *BufferSpaceUsed = ACPI_ROUND_UP_TO_NATIVE_WORD (
168                             (ACPI_SIZE) InternalObject->String.Length + 1);

170         ACPI_MEMCPY ((void *) DataSpace,
171             (void *) InternalObject->String.Pointer,
172             (ACPI_SIZE) InternalObject->String.Length + 1);
173         break;


175     case ACPI_TYPE_BUFFER:

177         ExternalObject->Buffer.Pointer = DataSpace;
178         ExternalObject->Buffer.Length  = InternalObject->Buffer.Length;
179         *BufferSpaceUsed = ACPI_ROUND_UP_TO_NATIVE_WORD (
180                             InternalObject->String.Length);

182         ACPI_MEMCPY ((void *) DataSpace,
183             (void *) InternalObject->Buffer.Pointer,
184             InternalObject->Buffer.Length);
185         break;


187     case ACPI_TYPE_INTEGER:

189         ExternalObject->Integer.Value = InternalObject->Integer.Value;
```



```
190         break;


192     case ACPI_TYPE_LOCAL_REFERENCE:

194         /* This is an object reference. */

196         switch (InternalObject->Reference.Class)
197         {
198         case ACPI_REFCLASS_NAME:

199             /*
200              * For namepath, return the object handle ("reference")
201              * We are referring to the namespace node
202              */
203             ExternalObject->Reference.Handle =
204                 InternalObject->Reference.Node;
205             ExternalObject->Reference.ActualType =
206                 AcpiNsGetType (InternalObject->Reference.Node);
207             break;

209         default:

211             /* All other reference types are unsupported */

213             return_ACPI_STATUS (AE_TYPE);
214         }
215         break;


217     case ACPI_TYPE_PROCESSOR:

219         ExternalObject->Processor.ProcId =
220             InternalObject->Processor.ProcId;
221         ExternalObject->Processor.PblkAddress =
222             InternalObject->Processor.Address;
223         ExternalObject->Processor.PblkLength =
224             InternalObject->Processor.Length;
225         break;


227     case ACPI_TYPE_POWER:

229         ExternalObject->PowerResource.SystemLevel =
230             InternalObject->PowerResource.SystemLevel;

232         ExternalObject->PowerResource.ResourceOrder =
233             InternalObject->PowerResource.ResourceOrder;
234         break;


236     default:
237         /*
238          * There is no corresponding external object type
239          */
240         ACPI_ERROR ((AE_INFO,
241             "Unsupported object type, cannot convert to external object: %s",
242             AcpiUtGetTypeName (InternalObject->Common.Type)));

244         return_ACPI_STATUS (AE_SUPPORT);
245     }

247     return_ACPI_STATUS (Status);
248 }
```

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtCopyIelementToEelement
 *
 * PARAMETERS:  ACPI_PKG_CALLBACK
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Copy one package element to another package element
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiUtCopyIelementToEelement (
    UINT8                   ObjectType,
    ACPI_OPERAND_OBJECT     *SourceObject,
    ACPI_GENERIC_STATE      *State,
    void                    *Context)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_PKG_INFO           *Info = (ACPI_PKG_INFO *) Context;
    ACPI_SIZE               ObjectSpace;
    UINT32                  ThisIndex;
    ACPI_OBJECT             *TargetObject;


    ACPI_FUNCTION_ENTRY ();


    ThisIndex    = State->Pkg.Index;
    TargetObject = (ACPI_OBJECT *)
        &((ACPI_OBJECT *)(State->Pkg.DestObject))->Package.Elements[ThisIndex];

    switch (ObjectType)
    {
    case ACPI_COPY_TYPE_SIMPLE:

        /*
         * This is a simple or null object
         */
        Status = AcpiUtCopyIsimpleToEsimple (SourceObject,
                    TargetObject, Info->FreeSpace, &ObjectSpace);
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }
        break;


    case ACPI_COPY_TYPE_PACKAGE:

        /*
         * Build the package object
         */
        TargetObject->Type              = ACPI_TYPE_PACKAGE;
        TargetObject->Package.Count     = SourceObject->Package.Count;
        TargetObject->Package.Elements  =
            ACPI_CAST_PTR (ACPI_OBJECT, Info->FreeSpace);

        /*
         * Pass the new package object back to the package walk routine
         */
        State->Pkg.ThisTargetObj = TargetObject;

        /*
         * Save space for the array of objects (Package elements)
```



```
         * update the buffer length counter
         */
        ObjectSpace = ACPI_ROUND_UP_TO_NATIVE_WORD (
                        (ACPI_SIZE) TargetObject->Package.Count *
                        sizeof (ACPI_OBJECT));
        break;

    default:

    default:
        return (AE_BAD_PARAMETER);
    }

    Info->FreeSpace   += ObjectSpace;
    Info->Length      += ObjectSpace;
    return (Status);
}
_____unchanged_portion_omitted_


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtCopyEsimpleToIsimple
 *
 * PARAMETERS:  ExternalObject      - The external object to be converted
 *              RetInternalObject   - Where the internal object is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: This function copies an external object to an internal one.
 *              NOTE: Pointers can be copied, we don't need to copy data.
 *              (The pointers have to be valid in our address space no matter
 *              what we do with them!)
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiUtCopyEsimpleToIsimple (
    ACPI_OBJECT             *ExternalObject,
    ACPI_OPERAND_OBJECT     **RetInternalObject)
{
    ACPI_OPERAND_OBJECT     *InternalObject;


    ACPI_FUNCTION_TRACE (UtCopyEsimpleToIsimple);


    /*
     * Simple types supported are: String, Buffer, Integer
     */
    switch (ExternalObject->Type)
    {
    case ACPI_TYPE_STRING:
    case ACPI_TYPE_BUFFER:
    case ACPI_TYPE_INTEGER:
    case ACPI_TYPE_LOCAL_REFERENCE:

        InternalObject = AcpiUtCreateInternalObject (
                            (UINT8) ExternalObject->Type);
        if (!InternalObject)
        {
            return_ACPI_STATUS (AE_NO_MEMORY);
        }
        break;

    case ACPI_TYPE_ANY: /* This is the case for a NULL object */
```

```
503         *RetInternalObject = NULL;
504         return_ACPI_STATUS (AE_OK);

506     default:

508         /* All other types are not supported */

510         ACPI_ERROR ((AE_INFO,
511             "Unsupported object type, cannot convert to internal object: %s",
512             AcpiUtGetTypeName (ExternalObject->Type)));

514         return_ACPI_STATUS (AE_SUPPORT);
515     }


518     /* Must COPY string and buffer contents */

520     switch (ExternalObject->Type)
521     {
522     case ACPI_TYPE_STRING:

524         InternalObject->String.Pointer =
525             ACPI_ALLOCATE_ZEROED ((ACPI_SIZE)
526                 ExternalObject->String.Length + 1);

528         if (!InternalObject->String.Pointer)
529         {
530             goto ErrorExit;
531         }

533         ACPI_MEMCPY (InternalObject->String.Pointer,
534                      ExternalObject->String.Pointer,
535                      ExternalObject->String.Length);

537         InternalObject->String.Length  = ExternalObject->String.Length;
538         break;


540     case ACPI_TYPE_BUFFER:

542         InternalObject->Buffer.Pointer =
543             ACPI_ALLOCATE_ZEROED (ExternalObject->Buffer.Length);
544         if (!InternalObject->Buffer.Pointer)
545         {
546             goto ErrorExit;
547         }

549         ACPI_MEMCPY (InternalObject->Buffer.Pointer,
550                      ExternalObject->Buffer.Pointer,
551                      ExternalObject->Buffer.Length);

553         InternalObject->Buffer.Length  = ExternalObject->Buffer.Length;

555         /* Mark buffer data valid */

557         InternalObject->Buffer.Flags |= AOPOBJ_DATA_VALID;
558         break;


560     case ACPI_TYPE_INTEGER:

562         InternalObject->Integer.Value   = ExternalObject->Integer.Value;
563         break;

565     case ACPI_TYPE_LOCAL_REFERENCE:
```



```
567         /* TBD: should validate incoming handle */

569         InternalObject->Reference.Class = ACPI_REFCLASS_NAME;
570         InternalObject->Reference.Node = ExternalObject->Reference.Handle;
571         break;

573     default:

575         /* Other types can't get here */

577         break;
578     }

580     *RetInternalObject = InternalObject;
581     return_ACPI_STATUS (AE_OK);


584 ErrorExit:
585     AcpiUtRemoveReference (InternalObject);
586     return_ACPI_STATUS (AE_NO_MEMORY);
587 }
```

*unchanged_portion_omitted_*

```
697 /******************************************************************************
698  *
699  * FUNCTION:    AcpiUtCopySimpleObject
700  *
701  * PARAMETERS:  SourceDesc          - The internal object to be copied
702  *              DestDesc            - New target object
703  *
704  * RETURN:      Status
705  *
706  * DESCRIPTION: Simple copy of one internal object to another. Reference count
707  *              of the destination object is preserved.
708  *
709  *****************************************************************************/

711 static ACPI_STATUS
712 AcpiUtCopySimpleObject (
713     ACPI_OPERAND_OBJECT     *SourceDesc,
714     ACPI_OPERAND_OBJECT     *DestDesc)
715 {
716     UINT16                  ReferenceCount;
717     ACPI_OPERAND_OBJECT     *NextObject;
718     ACPI_STATUS             Status;
719     ACPI_SIZE               CopySize;


722     /* Save fields from destination that we don't want to overwrite */

724     ReferenceCount = DestDesc->Common.ReferenceCount;
725     NextObject = DestDesc->Common.NextObject;

727     /*
728      * Copy the entire source object over the destination object.
729      * Note: Source can be either an operand object or namespace node.
730      */
731     CopySize = sizeof (ACPI_OPERAND_OBJECT);
732     if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc) == ACPI_DESC_TYPE_NAMED)
733     {
734         CopySize = sizeof (ACPI_NAMESPACE_NODE);
735     }

737     ACPI_MEMCPY (ACPI_CAST_PTR (char, DestDesc),
```

```
738         ACPI_CAST_PTR (char, SourceDesc), CopySize);

740     /* Restore the saved fields */

742     DestDesc->Common.ReferenceCount = ReferenceCount;
743     DestDesc->Common.NextObject = NextObject;

745     /* New object is not static, regardless of source */

747     DestDesc->Common.Flags &= ~AOPOBJ_STATIC_POINTER;

749     /* Handle the objects with extra data */

751     switch (DestDesc->Common.Type)
752     {
753     case ACPI_TYPE_BUFFER:
754         /*
755          * Allocate and copy the actual buffer if and only if:
756          * 1) There is a valid buffer pointer
757          * 2) The buffer has a length > 0
758          */
759         if ((SourceDesc->Buffer.Pointer) &&
760             (SourceDesc->Buffer.Length))
761         {
762             DestDesc->Buffer.Pointer =
763                 ACPI_ALLOCATE (SourceDesc->Buffer.Length);
764             if (!DestDesc->Buffer.Pointer)
765             {
766                 return (AE_NO_MEMORY);
767             }

769             /* Copy the actual buffer data */

771             ACPI_MEMCPY (DestDesc->Buffer.Pointer,
772                 SourceDesc->Buffer.Pointer, SourceDesc->Buffer.Length);
773         }
774         break;

776     case ACPI_TYPE_STRING:
777         /*
778          * Allocate and copy the actual string if and only if:
779          * 1) There is a valid string pointer
780          * (Pointer to a NULL string is allowed)
781          */
782         if (SourceDesc->String.Pointer)
783         {
784             DestDesc->String.Pointer =
785                 ACPI_ALLOCATE ((ACPI_SIZE) SourceDesc->String.Length + 1);
786             if (!DestDesc->String.Pointer)
787             {
788                 return (AE_NO_MEMORY);
789             }

791             /* Copy the actual string data */

793             ACPI_MEMCPY (DestDesc->String.Pointer, SourceDesc->String.Pointer,
794                 (ACPI_SIZE) SourceDesc->String.Length + 1);
795         }
796         break;

798     case ACPI_TYPE_LOCAL_REFERENCE:
799         /*
800          * We copied the reference object, so we now must add a reference
801          * to the object pointed to by the reference
802          *
803          * DDBHandle reference (from Load/LoadTable) is a special reference,
```



```
804          * it does not have a Reference.Object, so does not need to
805          * increase the reference count
806          */
807         if (SourceDesc->Reference.Class == ACPI_REFCLASS_TABLE)
808         {
809             break;
810         }

812         AcpiUtAddReference (SourceDesc->Reference.Object);
813         break;

815     case ACPI_TYPE_REGION:
816         /*
817          * We copied the Region Handler, so we now must add a reference
818          */
819         if (DestDesc->Region.Handler)
820         {
821             AcpiUtAddReference (DestDesc->Region.Handler);
822         }
823         break;

825     /*
826      * For Mutex and Event objects, we cannot simply copy the underlying
827      * OS object. We must create a new one.
828      */
829     case ACPI_TYPE_MUTEX:

831         Status = AcpiOsCreateMutex (&DestDesc->Mutex.OsMutex);
832         if (ACPI_FAILURE (Status))
833         {
834             return (Status);
835         }
836         break;

838     case ACPI_TYPE_EVENT:

840         Status = AcpiOsCreateSemaphore (ACPI_NO_UNIT_LIMIT, 0,
841                     &DestDesc->Event.OsSemaphore);
842         if (ACPI_FAILURE (Status))
843         {
844             return (Status);
845         }
846         break;

848     default:

850         /* Nothing to do for other simple objects */

852         break;
853     }

855     return (AE_OK);
856 }


859 /*******************************************************************************
860  *
861  * FUNCTION:    AcpiUtCopyIelementToIelement
862  *
863  * PARAMETERS:  ACPI_PKG_CALLBACK
864  *
865  * RETURN:      Status
866  *
867  * DESCRIPTION: Copy one package element to another package element
868  *
869  ******************************************************************************/
```

```
 871 static ACPI_STATUS
 872 AcpiUtCopyIelementToIelement (
 873     UINT8                   ObjectType,
 874     ACPI_OPERAND_OBJECT     *SourceObject,
 875     ACPI_GENERIC_STATE      *State,
 876     void                    *Context)
 877 {
 878     ACPI_STATUS             Status = AE_OK;
 879     UINT32                  ThisIndex;
 880     ACPI_OPERAND_OBJECT     **ThisTargetPtr;
 881     ACPI_OPERAND_OBJECT     *TargetObject;


 884     ACPI_FUNCTION_ENTRY ();


 887     ThisIndex     = State->Pkg.Index;
 888     ThisTargetPtr = (ACPI_OPERAND_OBJECT **)
 889                         &State->Pkg.DestObject->Package.Elements[ThisIndex];

 891     switch (ObjectType)
 892     {
 893     case ACPI_COPY_TYPE_SIMPLE:

 895         /* A null source object indicates a (legal) null package element */

 897         if (SourceObject)
 898         {
 899             /*
 900              * This is a simple object, just copy it
 901              */
 902             TargetObject = AcpiUtCreateInternalObject (
 903                                 SourceObject->Common.Type);
 904             if (!TargetObject)
 905             {
 906                 return (AE_NO_MEMORY);
 907             }

 909             Status = AcpiUtCopySimpleObject (SourceObject, TargetObject);
 910             if (ACPI_FAILURE (Status))
 911             {
 912                 goto ErrorExit;
 913             }

 915             *ThisTargetPtr = TargetObject;
 916         }
 917         else
 918         {
 919             /* Pass through a null element */

 921             *ThisTargetPtr = NULL;
 922         }
 923         break;


 925     case ACPI_COPY_TYPE_PACKAGE:

 926         /*
 927          * This object is a package - go down another nesting level
 928          * Create and build the package object
 929          */
 930         TargetObject = AcpiUtCreatePackageObject (SourceObject->Package.Count);
 931         if (!TargetObject)
 932         {
 933             return (AE_NO_MEMORY);
```



```
 934         }

 936         TargetObject->Common.Flags = SourceObject->Common.Flags;

 938         /* Pass the new package object back to the package walk routine */

 940         State->Pkg.ThisTargetObj = TargetObject;

 942         /* Store the object pointer in the parent package object */

 944         *ThisTargetPtr = TargetObject;
 945         break;

 947     default:

 957     default:
 949         return (AE_BAD_PARAMETER);
 950     }

 952     return (Status);

 954 ErrorExit:
 955     AcpiUtRemoveReference (TargetObject);
 956     return (Status);
 957 }
______unchanged_portion_omitted_
```

```
**********************************************************
   19080 Thu Dec 26 13:49:37 2013
new/usr/src/common/acpica/components/utilities/utdebug.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utdebug - Debug print/trace routines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTDEBUG_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utdebug")


#ifdef ACPI_DEBUG_OUTPUT

static ACPI_THREAD_ID       AcpiGbl_PrevThreadId = (ACPI_THREAD_ID) 0xFFFFFFFF;
static char                 *AcpiGbl_FnEntryStr = "----Entry";
```



```
static char                 *AcpiGbl_FnExitStr  = "----Exit-";

/* Local prototypes */

static const char *
AcpiUtTrimFunctionName (
    const char              *FunctionName);


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtInitStackPtrTrace
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Save the current CPU stack pointer at subsystem startup
 *
 ******************************************************************************/

void
AcpiUtInitStackPtrTrace (
    void)
{
    ACPI_SIZE               CurrentSp;


    AcpiGbl_EntryStackPointer = &CurrentSp;
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtTrackStackPtr
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Save the current CPU stack pointer
 *
 ******************************************************************************/

void
AcpiUtTrackStackPtr (
    void)
{
    ACPI_SIZE               CurrentSp;


    if (&CurrentSp < AcpiGbl_LowestStackPointer)
    {
        AcpiGbl_LowestStackPointer = &CurrentSp;
    }

    if (AcpiGbl_NestingLevel > AcpiGbl_DeepestNesting)
    {
        AcpiGbl_DeepestNesting = AcpiGbl_NestingLevel;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtTrimFunctionName
```

```
 *
 * PARAMETERS:  FunctionName        - Ascii string containing a procedure name
 *
 * RETURN:      Updated pointer to the function name
 *
 * DESCRIPTION: Remove the "Acpi" prefix from the function name, if present.
 *              This allows compiler macros such as __FUNCTION__ to be used
 *              with no change to the debug output.
 *
 ******************************************************************************/

static const char *
AcpiUtTrimFunctionName (
    const char              *FunctionName)
{

    /* All Function names are longer than 4 chars, check is safe */

    if (*(ACPI_CAST_PTR (UINT32, FunctionName)) == ACPI_PREFIX_MIXED)
    {
        /* This is the case where the original source has not been modified */

        return (FunctionName + 4);
    }

    if (*(ACPI_CAST_PTR (UINT32, FunctionName)) == ACPI_PREFIX_LOWER)
    {
        /* This is the case where the source has been 'linuxized' */

        return (FunctionName + 5);
    }

    return (FunctionName);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiDebugPrint
 *
 * PARAMETERS:  RequestedDebugLevel - Requested debug print level
 *              LineNumber          - Caller's line number (for error output)
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *              Format              - Printf format field
 *              ...                 - Optional printf arguments
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print error message with prefix consisting of the module name,
 *              line number, and component ID.
 *
 ******************************************************************************/

void  ACPI_INTERNAL_VAR_XFACE
AcpiDebugPrint (
    UINT32                  RequestedDebugLevel,
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
    const char              *Format,
    ...)
{
    ACPI_THREAD_ID          ThreadId;
```



```
    va_list                 args;


    /* Check if debug output enabled */

    if (!ACPI_IS_DEBUG_ENABLED (RequestedDebugLevel, ComponentId))
    {
        return;
    }

    /*
     * Thread tracking and context switch notification
     */
    ThreadId = AcpiOsGetThreadId ();
    if (ThreadId != AcpiGbl_PrevThreadId)
    {
        if (ACPI_LV_THREADS & AcpiDbgLevel)
        {
            AcpiOsPrintf (
                "\n**** Context Switch from TID %u to TID %u ****\n\n",
                (UINT32) AcpiGbl_PrevThreadId, (UINT32) ThreadId);
        }

        AcpiGbl_PrevThreadId = ThreadId;
        AcpiGbl_NestingLevel = 0;
    }

    /*
     * Display the module name, current line number, thread ID (if requested),
     * current procedure nesting level, and the current procedure name
     */
    AcpiOsPrintf ("%9s-%04ld ", ModuleName, LineNumber);

#ifdef ACPI_APPLICATION
    /*
     * For AcpiExec/iASL only, emit the thread ID and nesting level.
     * Note: nesting level is really only useful during a single-thread
     * execution. Otherwise, multiple threads will keep resetting the
     * level.
     */
    if (ACPI_LV_THREADS & AcpiDbgLevel)
    {
        AcpiOsPrintf ("[%u] ", (UINT32) ThreadId);
    }

    AcpiOsPrintf ("[%02ld] ", AcpiGbl_NestingLevel);
#endif

    AcpiOsPrintf ("%-22.22s: ", AcpiUtTrimFunctionName (FunctionName));

    va_start (args, Format);
    AcpiOsVprintf (Format, args);
    va_end (args);
}

ACPI_EXPORT_SYMBOL (AcpiDebugPrint)


/******************************************************************************
 *
 * FUNCTION:    AcpiDebugPrintRaw
 *
 * PARAMETERS:  RequestedDebugLevel - Requested debug print level
 *              LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
```

```
 *              ComponentId         - Caller's component ID
 *              Format              - Printf format field
 *              ...                 - Optional printf arguments
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print message with no headers. Has same interface as
 *              DebugPrint so that the same macros can be used.
 *
 ******************************************************************************/

void  ACPI_INTERNAL_VAR_XFACE
AcpiDebugPrintRaw (
    UINT32                  RequestedDebugLevel,
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
    const char              *Format,
    ...)
{
    va_list                 args;


    /* Check if debug output enabled */

    if (!ACPI_IS_DEBUG_ENABLED (RequestedDebugLevel, ComponentId))
    {
        return;
    }

    va_start (args, Format);
    AcpiOsVprintf (Format, args);
    va_end (args);
}

ACPI_EXPORT_SYMBOL (AcpiDebugPrintRaw)


/******************************************************************************
 *
 * FUNCTION:    AcpiUtTrace
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
 *              set in DebugLevel
 *
 ******************************************************************************/

void
AcpiUtTrace (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId)
{

    AcpiGbl_NestingLevel++;
    AcpiUtTrackStackPtr ();
```



```
    /* Check if enabled up-front for performance */

    if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
    {
        AcpiDebugPrint (ACPI_LV_FUNCTIONS,
            LineNumber, FunctionName, ModuleName, ComponentId,
            "%s\n", AcpiGbl_FnEntryStr);
    }
}

ACPI_EXPORT_SYMBOL (AcpiUtTrace)


/******************************************************************************
 *
 * FUNCTION:    AcpiUtTracePtr
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *              Pointer             - Pointer to display
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
 *              set in DebugLevel
 *
 ******************************************************************************/

void
AcpiUtTracePtr (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
    void                    *Pointer)
{

    AcpiGbl_NestingLevel++;
    AcpiUtTrackStackPtr ();

    /* Check if enabled up-front for performance */

    if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
    {
        AcpiDebugPrint (ACPI_LV_FUNCTIONS,
            LineNumber, FunctionName, ModuleName, ComponentId,
            "%s %p\n", AcpiGbl_FnEntryStr, Pointer);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtTraceStr
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *              String              - Additional string to display
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
```

```
 *              set in DebugLevel
 *
 ******************************************************************************/

void
AcpiUtTraceStr (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
    char                    *String)
{

    AcpiGbl_NestingLevel++;
    AcpiUtTrackStackPtr ();

    /* Check if enabled up-front for performance */

    if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
    {
        AcpiDebugPrint (ACPI_LV_FUNCTIONS,
            LineNumber, FunctionName, ModuleName, ComponentId,
            "%s %s\n", AcpiGbl_FnEntryStr, String);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtTraceU32
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *              Integer             - Integer to display
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
 *              set in DebugLevel
 *
 ******************************************************************************/

void
AcpiUtTraceU32 (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
    UINT32                  Integer)
{

    AcpiGbl_NestingLevel++;
    AcpiUtTrackStackPtr ();

    /* Check if enabled up-front for performance */

    if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
    {
        AcpiDebugPrint (ACPI_LV_FUNCTIONS,
            LineNumber, FunctionName, ModuleName, ComponentId,
            "%s %08X\n", AcpiGbl_FnEntryStr, Integer);
    }
}
```



```
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtExit
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
 *              set in DebugLevel
 *
 ******************************************************************************/

void
AcpiUtExit (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId)
{

    /* Check if enabled up-front for performance */

    if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
    {
        AcpiDebugPrint (ACPI_LV_FUNCTIONS,
            LineNumber, FunctionName, ModuleName, ComponentId,
            "%s\n", AcpiGbl_FnExitStr);
    }

    if (AcpiGbl_NestingLevel)
    {
        AcpiGbl_NestingLevel--;
    }
}

ACPI_EXPORT_SYMBOL (AcpiUtExit)


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStatusExit
 *
 * PARAMETERS:  LineNumber          - Caller's line number
 *              FunctionName        - Caller's procedure name
 *              ModuleName          - Caller's module name
 *              ComponentId         - Caller's component ID
 *              Status              - Exit status code
 *
 * RETURN:      None
 *
 * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
 *              set in DebugLevel. Prints exit status also.
 *
 ******************************************************************************/

void
AcpiUtStatusExit (
    UINT32                  LineNumber,
    const char              *FunctionName,
    const char              *ModuleName,
    UINT32                  ComponentId,
```

```
520     ACPI_STATUS             Status)
521 {
522
523     /* Check if enabled up-front for performance */
524
525     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
526     {
527         if (ACPI_SUCCESS (Status))
528         {
529             AcpiDebugPrint (ACPI_LV_FUNCTIONS,
530                 LineNumber, FunctionName, ModuleName, ComponentId,
531                 "%s %s\n", AcpiGbl_FnExitStr,
532                 AcpiFormatException (Status));
533         }
534         else
535         {
536             AcpiDebugPrint (ACPI_LV_FUNCTIONS,
537                 LineNumber, FunctionName, ModuleName, ComponentId,
538                 "%s ****Exception****: %s\n", AcpiGbl_FnExitStr,
539                 AcpiFormatException (Status));
540         }
541     }
542
543     if (AcpiGbl_NestingLevel)
544     {
545         AcpiGbl_NestingLevel--;
546     }
547 }
548
549 ACPI_EXPORT_SYMBOL (AcpiUtStatusExit)
550
551
552 /*******************************************************************************
553  *
554  * FUNCTION:    AcpiUtValueExit
555  *
556  * PARAMETERS:  LineNumber          - Caller's line number
557  *              FunctionName        - Caller's procedure name
558  *              ModuleName          - Caller's module name
559  *              ComponentId         - Caller's component ID
560  *              Value               - Value to be printed with exit msg
561  *
562  * RETURN:      None
563  *
564  * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
565  *              set in DebugLevel. Prints exit value also.
566  *
567  ******************************************************************************/
568
569 void
570 AcpiUtValueExit (
571     UINT32                  LineNumber,
572     const char              *FunctionName,
573     const char              *ModuleName,
574     UINT32                  ComponentId,
575     UINT64                  Value)
576 {
577
578     /* Check if enabled up-front for performance */
579
580     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
581     {
582         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
583             LineNumber, FunctionName, ModuleName, ComponentId,
584             "%s %8.8X%8.8X\n", AcpiGbl_FnExitStr,
585             ACPI_FORMAT_UINT64 (Value));
```



```
586     }
587
588     if (AcpiGbl_NestingLevel)
589     {
590         AcpiGbl_NestingLevel--;
591     }
592 }
593
594 ACPI_EXPORT_SYMBOL (AcpiUtValueExit)
595
596
597 /*******************************************************************************
598  *
599  * FUNCTION:    AcpiUtPtrExit
600  *
601  * PARAMETERS:  LineNumber          - Caller's line number
602  *              FunctionName        - Caller's procedure name
603  *              ModuleName          - Caller's module name
604  *              ComponentId         - Caller's component ID
605  *              Ptr                 - Pointer to display
606  *
607  * RETURN:      None
608  *
609  * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
610  *              set in DebugLevel. Prints exit value also.
611  *
612  ******************************************************************************/
613
614 void
615 AcpiUtPtrExit (
616     UINT32                  LineNumber,
617     const char              *FunctionName,
618     const char              *ModuleName,
619     UINT32                  ComponentId,
620     UINT8                   *Ptr)
621 {
622
623     /* Check if enabled up-front for performance */
624
625     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
626     {
627         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
628             LineNumber, FunctionName, ModuleName, ComponentId,
629             "%s %p\n", AcpiGbl_FnExitStr, Ptr);
630     }
631
632     if (AcpiGbl_NestingLevel)
633     {
634         AcpiGbl_NestingLevel--;
635     }
636 }
637
638 #endif
```

```
**********************************************************
   16487 Thu Dec 26 13:49:37 2013
new/usr/src/common/acpica/components/utilities/utdecode.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utdecode - Utility decoding routines (value-to-string)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTDECODE_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acnamesp.h"

  50 #define _COMPONENT          ACPI_UTILITIES
  51          ACPI_MODULE_NAME    ("utdecode")


  54 /******************************************************************************
  55  *
  56  * FUNCTION:    AcpiFormatException
  57  *
  58  * PARAMETERS:  Status       - The ACPI_STATUS code to be formatted
  59  *
```



```
  60  * RETURN:      A string containing the exception text. A valid pointer is
  61  *              always returned.
  62  *
  63  * DESCRIPTION: This function translates an ACPI exception into an ASCII string
  64  *              It is here instead of utxface.c so it is always present.
  65  *
  66  *****************************************************************************/

  68 const char *
  69 AcpiFormatException (
  70     ACPI_STATUS             Status)
  71 {
  72     const char              *Exception = NULL;


  75     ACPI_FUNCTION_ENTRY ();


  78     Exception = AcpiUtValidateException (Status);
  79     if (!Exception)
  80     {
  81         /* Exception code was not recognized */

  83         ACPI_ERROR ((AE_INFO,
  84             "Unknown exception code: 0x%8.8X", Status));

  86         Exception = "UNKNOWN_STATUS_CODE";
  87     }

  89     return (ACPI_CAST_PTR (const char, Exception));
  90 }

  92 ACPI_EXPORT_SYMBOL (AcpiFormatException)


  54 /*
  55  * Properties of the ACPI Object Types, both internal and external.
  56  * The table is indexed by values of ACPI_OBJECT_TYPE
  57  */
  58 const UINT8                     AcpiGbl_NsProperties[ACPI_NUM_NS_TYPES] =
  59 {
  60     ACPI_NS_NORMAL,                     /* 00 Any              */
  61     ACPI_NS_NORMAL,                     /* 01 Number           */
  62     ACPI_NS_NORMAL,                     /* 02 String           */
  63     ACPI_NS_NORMAL,                     /* 03 Buffer           */
  64     ACPI_NS_NORMAL,                     /* 04 Package          */
  65     ACPI_NS_NORMAL,                     /* 05 FieldUnit        */
  66     ACPI_NS_NEWSCOPE,                   /* 06 Device           */
  67     ACPI_NS_NORMAL,                     /* 07 Event            */
  68     ACPI_NS_NEWSCOPE,                   /* 08 Method           */
  69     ACPI_NS_NORMAL,                     /* 09 Mutex            */
  70     ACPI_NS_NORMAL,                     /* 10 Region           */
  71     ACPI_NS_NEWSCOPE,                   /* 11 Power            */
  72     ACPI_NS_NEWSCOPE,                   /* 12 Processor        */
  73     ACPI_NS_NEWSCOPE,                   /* 13 Thermal          */
  74     ACPI_NS_NORMAL,                     /* 14 BufferField      */
  75     ACPI_NS_NORMAL,                     /* 15 DdbHandle        */
  76     ACPI_NS_NORMAL,                     /* 16 Debug Object     */
  77     ACPI_NS_NORMAL,                     /* 17 DefField         */
  78     ACPI_NS_NORMAL,                     /* 18 BankField        */
  79     ACPI_NS_NORMAL,                     /* 19 IndexField       */
  80     ACPI_NS_NORMAL,                     /* 20 Reference        */
  81     ACPI_NS_NORMAL,                     /* 21 Alias            */
  82     ACPI_NS_NORMAL,                     /* 22 MethodAlias      */
  83     ACPI_NS_NORMAL,                     /* 23 Notify           */
  84     ACPI_NS_NORMAL,                     /* 24 Address Handler  */
```

```
    ACPI_NS_NEWSCOPE | ACPI_NS_LOCAL,   /* 25 Resource Desc    */
    ACPI_NS_NEWSCOPE | ACPI_NS_LOCAL,   /* 26 Resource Field   */
    ACPI_NS_NEWSCOPE,                   /* 27 Scope            */
    ACPI_NS_NORMAL,                     /* 28 Extra            */
    ACPI_NS_NORMAL,                     /* 29 Data             */
    ACPI_NS_NORMAL                      /* 30 Invalid          */
};
```

*unchanged_portion_omitted_*

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtGetRegionName
 *
 * PARAMETERS:  Space ID            - ID for the region
 *
 * RETURN:      Decoded region SpaceId name
 *
 * DESCRIPTION: Translate a Space ID into a name string (Debug only)
 *
 ******************************************************************************/

/* Region type decoding */

const char        *AcpiGbl_RegionTypes[ACPI_NUM_PREDEFINED_REGIONS] =
{
    "SystemMemory",     /* 0x00 */
    "SystemIO",         /* 0x01 */
    "PCI_Config",       /* 0x02 */
    "EmbeddedControl",  /* 0x03 */
    "SMBus",            /* 0x04 */
    "SystemCMOS",       /* 0x05 */
    "PCIBARTarget",     /* 0x06 */
    "IPMI",             /* 0x07 */
    "GeneralPurposeIo", /* 0x08 */
    "GenericSerialBus", /* 0x09 */
    "PCC"               /* 0x0A */
    "SystemMemory",
    "SystemIO",
    "PCI_Config",
    "EmbeddedControl",
    "SMBus",
    "SystemCMOS",
    "PCIBARTarget",
    "IPMI"
};
```

*unchanged_portion_omitted_*

```
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtGetNotifyName
 *
 * PARAMETERS:  NotifyValue     - Value from the Notify() request
 *
 * RETURN:      Decoded name for the notify value
 *
 * DESCRIPTION: Translate a Notify Value to a notify namestring.
 *
 ******************************************************************************/

/* Names for Notify() values, used for debug output */

static const char           *AcpiGbl_NotifyValueNames[ACPI_NOTIFY_MAX + 1] =
static const char           *AcpiGbl_NotifyValueNames[] =
{
```



```
    /* 00 */ "Bus Check",
    /* 01 */ "Device Check",
    /* 02 */ "Device Wake",
    /* 03 */ "Eject Request",
    /* 04 */ "Device Check Light",
    /* 05 */ "Frequency Mismatch",
    /* 06 */ "Bus Mode Mismatch",
    /* 07 */ "Power Fault",
    /* 08 */ "Capabilities Check",
    /* 09 */ "Device PLD Check",
    /* 10 */ "Reserved",
    /* 11 */ "System Locality Update",
    /* 12 */ "Shutdown Request"
    "Bus Check",
    "Device Check",
    "Device Wake",
    "Eject Request",
    "Device Check Light",
    "Frequency Mismatch",
    "Bus Mode Mismatch",
    "Power Fault",
    "Capabilities Check",
    "Device PLD Check",
    "Reserved",
    "System Locality Update"
};

const char *
AcpiUtGetNotifyName (
    UINT32                  NotifyValue)
{

    if (NotifyValue <= ACPI_NOTIFY_MAX)
    {
        return (AcpiGbl_NotifyValueNames[NotifyValue]);
    }
    else if (NotifyValue <= ACPI_MAX_SYS_NOTIFY)
    {
        return ("Reserved");
    }
    else if (NotifyValue <= ACPI_MAX_DEVICE_SPECIFIC_NOTIFY)
    else /* Greater or equal to 0x80 */
    {
        return ("Device Specific");
        return ("**Device Specific**");
    }
    else
    {
        return ("Hardware Specific");
    }
}
```

*unchanged_portion_omitted_*

****
23081 Thu Dec 26 13:49:38 2013
new/usr/src/common/acpica/components/utilities/utdelete.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: utdelete - object deletion and reference count utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTDELETE_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"
#include "acnamesp.h"
#include "acevents.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utdelete")

/* Local prototypes */

static void
AcpiUtDeleteInternalObj (
```



```
    ACPI_OPERAND_OBJECT     *Object);

static void
AcpiUtUpdateRefCount (
    ACPI_OPERAND_OBJECT     *Object,
    UINT32                  Action);


/******************************************************************************
 *
 * FUNCTION:    AcpiUtDeleteInternalObj
 *
 * PARAMETERS:  Object          - Object to be deleted
 *
 * RETURN:      None
 *
 * DESCRIPTION: Low level object deletion, after reference counts have been
 *              updated (All reference counts, including sub-objects!)
 *
 *****************************************************************************/

static void
AcpiUtDeleteInternalObj (
    ACPI_OPERAND_OBJECT     *Object)
{
    void                    *ObjPointer = NULL;
    ACPI_OPERAND_OBJECT     *HandlerDesc;
    ACPI_OPERAND_OBJECT     *SecondDesc;
    ACPI_OPERAND_OBJECT     *NextDesc;
    ACPI_OPERAND_OBJECT     **LastObjPtr;


    ACPI_FUNCTION_TRACE_PTR (UtDeleteInternalObj, Object);


    if (!Object)
    {
        return_VOID;
    }

    /*
     * Must delete or free any pointers within the object that are not
     * actual ACPI objects (for example, a raw buffer pointer).
     */
    switch (Object->Common.Type)
    {
    case ACPI_TYPE_STRING:

        ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "**** String %p, ptr %p\n",
            Object, Object->String.Pointer));

        /* Free the actual string buffer */

        if (!(Object->Common.Flags & AOPOBJ_STATIC_POINTER))
        {
            /* But only if it is NOT a pointer into an ACPI table */

            ObjPointer = Object->String.Pointer;
        }
        break;


    case ACPI_TYPE_BUFFER:

        ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "**** Buffer %p, ptr %p\n",
            Object, Object->Buffer.Pointer));
```

```
126         /* Free the actual buffer */
128         if (!(Object->Common.Flags & AOPOBJ_STATIC_POINTER))
129         {
130             /* But only if it is NOT a pointer into an ACPI table */
132             ObjPointer = Object->Buffer.Pointer;
133         }
134         break;
136     case ACPI_TYPE_PACKAGE:
138         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, " **** Package of count %X\n",
139             Object->Package.Count));
141         /*
142          * Elements of the package are not handled here, they are deleted
143          * separately
144          */
146         /* Free the (variable length) element pointer array */
148         ObjPointer = Object->Package.Elements;
149         break;
151     /*
152      * These objects have a possible list of notify handlers.
153      * Device object also may have a GPE block.
154      */
155     case ACPI_TYPE_DEVICE:
157         if (Object->Device.GpeBlock)
158         {
159             (void) AcpiEvDeleteGpeBlock (Object->Device.GpeBlock);
160         }
162         /*lint -fallthrough */
164     case ACPI_TYPE_PROCESSOR:
165     case ACPI_TYPE_THERMAL:
167         /* Walk the address handler list for this object */
170         /* Walk the notify handler list for this object */
169         HandlerDesc = Object->CommonNotify.Handler;
170         while (HandlerDesc)
171         {
172             NextDesc = HandlerDesc->AddressSpace.Next;
173             AcpiUtRemoveReference (HandlerDesc);
174             HandlerDesc = NextDesc;
175         }
176         break;
178     case ACPI_TYPE_MUTEX:
180         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
181             "***** Mutex %p, OS Mutex %p\n",
182             Object, Object->Mutex.OsMutex));
184         if (Object == AcpiGbl_GlobalLockMutex)
185         {
186             /* Global Lock has extra semaphore */
```



```
188             (void) AcpiOsDeleteSemaphore (AcpiGbl_GlobalLockSemaphore);
189             AcpiGbl_GlobalLockSemaphore = NULL;
191             AcpiOsDeleteMutex (Object->Mutex.OsMutex);
192             AcpiGbl_GlobalLockMutex = NULL;
193         }
194         else
195         {
196             AcpiExUnlinkMutex (Object);
197             AcpiOsDeleteMutex (Object->Mutex.OsMutex);
198         }
199         break;
201     case ACPI_TYPE_EVENT:
203         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
204             "***** Event %p, OS Semaphore %p\n",
205             Object, Object->Event.OsSemaphore));
207         (void) AcpiOsDeleteSemaphore (Object->Event.OsSemaphore);
208         Object->Event.OsSemaphore = NULL;
209         break;
211     case ACPI_TYPE_METHOD:
213         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
214             "***** Method %p\n", Object));
216         /* Delete the method mutex if it exists */
218         if (Object->Method.Mutex)
219         {
220             AcpiOsDeleteMutex (Object->Method.Mutex->Mutex.OsMutex);
221             AcpiUtDeleteObjectDesc (Object->Method.Mutex);
222             Object->Method.Mutex = NULL;
223         }
224         break;
226     case ACPI_TYPE_REGION:
228         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
229             "***** Region %p\n", Object));
231         /*
232          * Update AddressRange list. However, only permanent regions
233          * are installed in this list. (Not created within a method)
234          */
235         if (!(Object->Region.Node->Flags & ANOBJ_TEMPORARY))
236         {
237             AcpiUtRemoveAddressRange (Object->Region.SpaceId,
238                 Object->Region.Node);
239         }
241         SecondDesc = AcpiNsGetSecondaryObject (Object);
242         if (SecondDesc)
243         {
244             /*
245              * Free the RegionContext if and only if the handler is one of the
246              * default handlers -- and therefore, we created the context object
247              * locally, it was not created by an external caller.
248              */
249             HandlerDesc = Object->Region.Handler;
```

```
250             if (HandlerDesc)
251             {
252                 NextDesc = HandlerDesc->AddressSpace.RegionList;
253                 LastObjPtr = &HandlerDesc->AddressSpace.RegionList;

255                 /* Remove the region object from the handler's list */

257                 while (NextDesc)
258                 {
259                     if (NextDesc == Object)
260                     {
261                         *LastObjPtr = NextDesc->Region.Next;
262                         break;
263                     }

265                     /* Walk the linked list of handler */

267                     LastObjPtr = &NextDesc->Region.Next;
268                     NextDesc = NextDesc->Region.Next;
269                 }

271                 if (HandlerDesc->AddressSpace.HandlerFlags &
272                     ACPI_ADDR_HANDLER_DEFAULT_INSTALLED)
273                 {
274                     /* Deactivate region and free region context */

276                     if (HandlerDesc->AddressSpace.Setup)
277                     {
278                         (void) HandlerDesc->AddressSpace.Setup (Object,
279                             ACPI_REGION_DEACTIVATE,
280                             HandlerDesc->AddressSpace.Context,
281                             &SecondDesc->Extra.RegionContext);
282                     }
283                 }

285                 AcpiUtRemoveReference (HandlerDesc);
286             }

288             /* Now we can free the Extra object */

290             AcpiUtDeleteObjectDesc (SecondDesc);
291         }
292         break;


294     case ACPI_TYPE_BUFFER_FIELD:

296         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
297             "***** Buffer Field %p\n", Object));

299         SecondDesc = AcpiNsGetSecondaryObject (Object);
300         if (SecondDesc)
301         {
302             AcpiUtDeleteObjectDesc (SecondDesc);
303         }
304         break;


306     case ACPI_TYPE_LOCAL_BANK_FIELD:

308         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
309             "***** Bank Field %p\n", Object));

311         SecondDesc = AcpiNsGetSecondaryObject (Object);
312         if (SecondDesc)
313         {
```



```
314             AcpiUtDeleteObjectDesc (SecondDesc);
315         }
316         break;

318     default:

318     default:
320         break;
321     }

323     /* Free any allocated memory (pointer within the object) found above */

325     if (ObjPointer)
326     {
327         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Deleting Object Subptr %p\n",
328             ObjPointer));
329         ACPI_FREE (ObjPointer);
330     }

332     /* Now the object can be safely deleted */

334     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Deleting Object %p [%s]\n",
335         Object, AcpiUtGetObjectTypeName (Object)));

337     AcpiUtDeleteObjectDesc (Object);
338     return_VOID;
339 }


342 /*******************************************************************************
343  *
344  * FUNCTION:    AcpiUtDeleteInternalObjectList
345  *
346  * PARAMETERS:  ObjList         - Pointer to the list to be deleted
347  *
348  * RETURN:      None
349  *
350  * DESCRIPTION: This function deletes an internal object list, including both
351  *              simple objects and package objects
352  *
353  ******************************************************************************/

355 void
356 AcpiUtDeleteInternalObjectList (
357     ACPI_OPERAND_OBJECT     **ObjList)
358 {
359     ACPI_OPERAND_OBJECT     **InternalObj;


362     ACPI_FUNCTION_ENTRY ();
361     ACPI_FUNCTION_TRACE (UtDeleteInternalObjectList);


365     /* Walk the null-terminated internal list */

367     for (InternalObj = ObjList; *InternalObj; InternalObj++)
368     {
369         AcpiUtRemoveReference (*InternalObj);
370     }

372     /* Free the combined parameter pointer list and object array */

374     ACPI_FREE (ObjList);
375     return;
374     return_VOID;
376 }
```

```
 379 /*******************************************************************************
 380  *
 381  * FUNCTION:    AcpiUtUpdateRefCount
 382  *
 383  * PARAMETERS:  Object          - Object whose ref count is to be updated
 384  *              Action          - What to do (REF_INCREMENT or REF_DECREMENT)
 383  *              Action          - What to do
 385  *
 386  * RETURN:      None. Sets new reference count within the object
 385  * RETURN:      New ref count
 387  *
 388  * DESCRIPTION: Modify the reference count for an internal acpi object
 387  * DESCRIPTION: Modify the ref count and return it.
 389  *
 390  ******************************************************************************/

 392 static void
 393 AcpiUtUpdateRefCount (
 394     ACPI_OPERAND_OBJECT     *Object,
 395     UINT32                  Action)
 396 {
 397     UINT16                  OriginalCount;
 398     UINT16                  NewCount = 0;
 399     ACPI_CPU_FLAGS          LockFlags;
 396     UINT16                  Count;
 397     UINT16                  NewCount;


 402     ACPI_FUNCTION_NAME (UtUpdateRefCount);


 405     if (!Object)
 406     {
 407         return;
 408     }

 408     Count = Object->Common.ReferenceCount;
 409     NewCount = Count;

 410     /*
 411      * Always get the reference count lock. Note: Interpreter and/or
 412      * Namespace is not always locked when this function is called.
 412      * Perform the reference count action (increment, decrement, force delete)
 413      */
 414     LockFlags = AcpiOsAcquireLock (AcpiGbl_ReferenceCountLock);
 415     OriginalCount = Object->Common.ReferenceCount;

 417     /* Perform the reference count action (increment, decrement) */

 419     switch (Action)
 420     {
 421     case REF_INCREMENT:

 423         NewCount = OriginalCount + 1;
 418         NewCount++;
 424         Object->Common.ReferenceCount = NewCount;
 425         AcpiOsReleaseLock (AcpiGbl_ReferenceCountLock, LockFlags);

 427         /* The current reference count should never be zero here */

 429         if (!OriginalCount)
 430         {
 431             ACPI_WARNING ((AE_INFO,
 432                 "Obj %p, Reference Count was zero before increment\n",
```



```
 433                 Object));
 434         }

 436         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 437             "Obj %p Type %.2X Refs %.2X [Incremented]\n",
 438             Object, Object->Common.Type, NewCount));
 422             "Obj %p Refs=%X, [Incremented]\n",
 423             Object, NewCount));
 439         break;

 441     case REF_DECREMENT:

 443         /* The current reference count must be non-zero */
 428         if (Count < 1)
 429         {
 430             ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 431                 "Obj %p Refs=%X, can't decrement! (Set to 0)\n",
 432                 Object, NewCount));

 445         if (OriginalCount)
 434             NewCount = 0;
 435         }
 436         else
 446         {
 447             NewCount = OriginalCount - 1;
 448             Object->Common.ReferenceCount = NewCount;
 438             NewCount--;

 440             ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 441                 "Obj %p Refs=%X, [Decremented]\n",
 442                 Object, NewCount));
 449         }

 451         AcpiOsReleaseLock (AcpiGbl_ReferenceCountLock, LockFlags);

 453         if (!OriginalCount)
 445         if (Object->Common.Type == ACPI_TYPE_METHOD)
 454         {
 455             ACPI_WARNING ((AE_INFO,
 456                 "Obj %p, Reference Count is already zero, cannot decrement\n",
 457                 Object));
 447             ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 448                 "Method Obj %p Refs=%X, [Decremented]\n", Object, NewCount));
 458         }

 460         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 461             "Obj %p Type %.2X Refs %.2X [Decremented]\n",
 462             Object, Object->Common.Type, NewCount));

 464         /* Actually delete the object on a reference count of zero */

 451         Object->Common.ReferenceCount = NewCount;
 466         if (NewCount == 0)
 467         {
 468             AcpiUtDeleteInternalObj (Object);
 469         }
 470         break;

 458     case REF_FORCE_DELETE:

 460         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 461             "Obj %p Refs=%X, Force delete! (Set to 0)\n", Object, Count));

 463         NewCount = 0;
 464         Object->Common.ReferenceCount = NewCount;
 465         AcpiUtDeleteInternalObj (Object);
```

```
466         break;

472     default:

474         AcpiOsReleaseLock (AcpiGbl_ReferenceCountLock, LockFlags);
475         ACPI_ERROR ((AE_INFO, "Unknown Reference Count action (0x%X)",
476             Action));
477         return;
470         ACPI_ERROR ((AE_INFO, "Unknown action (0x%X)", Action));
471         break;
478     }

480     /*
481      * Sanity check the reference count, for debug purposes only.
482      * (A deleted object will have a huge reference count)
483      */
484     if (NewCount > ACPI_MAX_REFERENCE_COUNT)
478     if (Count > ACPI_MAX_REFERENCE_COUNT)
485     {
486         ACPI_WARNING ((AE_INFO,
487             "Large Reference Count (0x%X) in object %p, Type=0x%.2X",
488             NewCount, Object, Object->Common.Type));
481             "Large Reference Count (0x%X) in object %p", Count, Object));
489     }
490 }


493 /*******************************************************************************
494  *
495  * FUNCTION:    AcpiUtUpdateObjectReference
496  *
497  * PARAMETERS:  Object              - Increment ref count for this object
498  *                                    and all sub-objects
499  *              Action              - Either REF_INCREMENT or REF_DECREMENT
492  *              Action              - Either REF_INCREMENT or REF_DECREMENT or
493  *                                    REF_FORCE_DELETE
500  *
501  * RETURN:      Status
502  *
503  * DESCRIPTION: Increment the object reference count
504  *
505  * Object references are incremented when:
506  * 1) An object is attached to a Node (namespace object)
507  * 2) An object is copied (all subobjects must be incremented)
508  *
509  * Object references are decremented when:
510  * 1) An object is detached from an Node
511  *
512  ******************************************************************************/

514 ACPI_STATUS
515 AcpiUtUpdateObjectReference (
516     ACPI_OPERAND_OBJECT     *Object,
517     UINT16                  Action)
518 {
519     ACPI_STATUS             Status = AE_OK;
520     ACPI_GENERIC_STATE      *StateList = NULL;
521     ACPI_OPERAND_OBJECT     *NextObject = NULL;
522     ACPI_OPERAND_OBJECT     *PrevObject;
523     ACPI_GENERIC_STATE      *State;
524     UINT32                  i;


527     ACPI_FUNCTION_NAME (UtUpdateObjectReference);
520     ACPI_FUNCTION_TRACE_PTR (UtUpdateObjectReference, Object);
```



```
530     while (Object)
531     {
532         /* Make sure that this isn't a namespace handle */

534         if (ACPI_GET_DESCRIPTOR_TYPE (Object) == ACPI_DESC_TYPE_NAMED)
535         {
536             ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
537                 "Object %p is NS handle\n", Object));
538             return (AE_OK);
531             return_ACPI_STATUS (AE_OK);
539         }

541         /*
542          * All sub-objects must have their reference count incremented also.
543          * Different object types have different subobjects.
544          */
545         switch (Object->Common.Type)
546         {
547         case ACPI_TYPE_DEVICE:
548         case ACPI_TYPE_PROCESSOR:
549         case ACPI_TYPE_POWER:
550         case ACPI_TYPE_THERMAL:
551             /*
552              * Update the notify objects for these types (if present)
553              * Two lists, system and device notify handlers.
554              */
555             for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
556             {
557                 PrevObject = Object->CommonNotify.NotifyList[i];
558                 while (PrevObject)
559                 {
560                     NextObject = PrevObject->Notify.Next[i];
561                     AcpiUtUpdateRefCount (PrevObject, Action);
562                     PrevObject = NextObject;
563                 }
564             }

545             /* Update the notify objects for these types (if present) */

547             AcpiUtUpdateRefCount (Object->CommonNotify.SystemNotify, Action);
548             AcpiUtUpdateRefCount (Object->CommonNotify.DeviceNotify, Action);
565             break;

567         case ACPI_TYPE_PACKAGE:
568             /*
569              * We must update all the sub-objects of the package,
570              * each of whom may have their own sub-objects.
571              */
572             for (i = 0; i < Object->Package.Count; i++)
573             {
574                 /*
575                  * Null package elements are legal and can be simply
576                  * ignored.
559                  * Push each element onto the stack for later processing.
560                  * Note: There can be null elements within the package,
561                  * these are simply ignored
577                  */
578                 NextObject = Object->Package.Elements[i];
579                 if (!NextObject)
580                 {
581                     continue;
582                 }

584                 switch (NextObject->Common.Type)
585                 {
```

```
586                 case ACPI_TYPE_INTEGER:
587                 case ACPI_TYPE_STRING:
588                 case ACPI_TYPE_BUFFER:
589                     /*
590                      * For these very simple sub-objects, we can just
591                      * update the reference count here and continue.
592                      * Greatly increases performance of this operation.
593                      */
594                     AcpiUtUpdateRefCount (NextObject, Action);
595                     break;

597                 default:
598                     /*
599                      * For complex sub-objects, push them onto the stack
600                      * for later processing (this eliminates recursion.)
601                      */
602                     Status = AcpiUtCreateUpdateStateAndPush (
603                                 NextObject, Action, &StateList);
564                              Object->Package.Elements[i], Action, &StateList);
604                     if (ACPI_FAILURE (Status))
605                     {
606                         goto ErrorExit;
607                     }
608                     break;
609                 }
610             }
611             NextObject = NULL;
612             break;

614         case ACPI_TYPE_BUFFER_FIELD:

616             NextObject = Object->BufferField.BufferObj;
617             break;

619         case ACPI_TYPE_LOCAL_REGION_FIELD:

621             NextObject = Object->Field.RegionObj;
622             break;

624         case ACPI_TYPE_LOCAL_BANK_FIELD:

626             NextObject = Object->BankField.BankObj;
627             Status = AcpiUtCreateUpdateStateAndPush (
628                         Object->BankField.RegionObj, Action, &StateList);
629             if (ACPI_FAILURE (Status))
630             {
631                 goto ErrorExit;
632             }
633             break;

635         case ACPI_TYPE_LOCAL_INDEX_FIELD:

637             NextObject = Object->IndexField.IndexObj;
638             Status = AcpiUtCreateUpdateStateAndPush (
639                         Object->IndexField.DataObj, Action, &StateList);
640             if (ACPI_FAILURE (Status))
641             {
642                 goto ErrorExit;
643             }
644             break;

646         case ACPI_TYPE_LOCAL_REFERENCE:
647             /*
648              * The target of an Index (a package, string, or buffer) or a named
649              * reference must track changes to the ref count of the index or
650              * target object.
```



```
651              */
652             if ((Object->Reference.Class == ACPI_REFCLASS_INDEX) ||
653                 (Object->Reference.Class== ACPI_REFCLASS_NAME))
654             {
655                 NextObject = Object->Reference.Object;
656             }
657             break;

659         case ACPI_TYPE_REGION:
660         default:

662             break; /* No subobjects for all other types */
663         }

665         /*
666          * Now we can update the count in the main object. This can only
667          * happen after we update the sub-objects in case this causes the
668          * main object to be deleted.
669          */
670         AcpiUtUpdateRefCount (Object, Action);
671         Object = NULL;

673         /* Move on to the next object to be updated */

675         if (NextObject)
676         {
677             Object = NextObject;
678             NextObject = NULL;
679         }
680         else if (StateList)
681         {
682             State = AcpiUtPopGenericState (&StateList);
683             Object = State->Update.Object;
684             AcpiUtDeleteGenericState (State);
685         }
686     }

688     return (AE_OK);
645     return_ACPI_STATUS (AE_OK);


691 ErrorExit:

693     ACPI_EXCEPTION ((AE_INFO, Status,
694         "Could not update object reference count"));

696     /* Free any stacked Update State objects */

698     while (StateList)
699     {
700         State = AcpiUtPopGenericState (&StateList);
701         AcpiUtDeleteGenericState (State);
702     }

704     return (Status);
661     return_ACPI_STATUS (Status);
705 }


708 /*******************************************************************************
709  *
710  * FUNCTION:    AcpiUtAddReference
711  *
712  * PARAMETERS:  Object          - Object whose reference count is to be
713  *                                incremented
714  *
```

```
 715  * RETURN:      None
 716  *
 717  * DESCRIPTION: Add one reference to an ACPI object
 718  *
 719  ******************************************************************************/

 721 void
 722 AcpiUtAddReference (
 723     ACPI_OPERAND_OBJECT     *Object)
 724 {

 726     ACPI_FUNCTION_NAME (UtAddReference);
 683     ACPI_FUNCTION_TRACE_PTR (UtAddReference, Object);


 729     /* Ensure that we have a valid object */

 731     if (!AcpiUtValidInternalObject (Object))
 732     {
 733         return;
 690         return_VOID;
 734     }

 736     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 737         "Obj %p Current Refs=%X [To Be Incremented]\n",
 738         Object, Object->Common.ReferenceCount));

 740     /* Increment the reference count */

 742     (void) AcpiUtUpdateObjectReference (Object, REF_INCREMENT);
 743     return;
 700     return_VOID;
 744 }


 747 /******************************************************************************
 748  *
 749  * FUNCTION:    AcpiUtRemoveReference
 750  *
 751  * PARAMETERS:  Object          - Object whose ref count will be decremented
 752  *
 753  * RETURN:      None
 754  *
 755  * DESCRIPTION: Decrement the reference count of an ACPI internal object
 756  *
 757  ******************************************************************************/

 759 void
 760 AcpiUtRemoveReference (
 761     ACPI_OPERAND_OBJECT     *Object)
 762 {

 764     ACPI_FUNCTION_NAME (UtRemoveReference);
 721     ACPI_FUNCTION_TRACE_PTR (UtRemoveReference, Object);


 767     /*
 768      * Allow a NULL pointer to be passed in, just ignore it. This saves
 769      * each caller from having to check. Also, ignore NS nodes.
 727      *
 770      */
 771     if (!Object ||
 772         (ACPI_GET_DESCRIPTOR_TYPE (Object) == ACPI_DESC_TYPE_NAMED))

 774     {
 775         return;
```



```
 733         return_VOID;
 776     }

 778     /* Ensure that we have a valid object */

 780     if (!AcpiUtValidInternalObject (Object))
 781     {
 782         return;
 740         return_VOID;
 783     }

 785     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
 786         "Obj %p Current Refs=%X [To Be Decremented]\n",
 787         Object, Object->Common.ReferenceCount));

 789     /*
 790      * Decrement the reference count, and only actually delete the object
 791      * if the reference count becomes 0. (Must also decrement the ref count
 792      * of all subobjects!)
 793      */
 794     (void) AcpiUtUpdateObjectReference (Object, REF_DECREMENT);
 795     return;
 753     return_VOID;
 796 }
```

```
**********************************************************
   10595 Thu Dec 26 13:49:38 2013
new/usr/src/common/acpica/components/utilities/uterror.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: uterror - Various internal error/warning output functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTERROR_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("uterror")


/*
 * This module contains internal error functions that may
 * be configured out.
 */
#if !defined (ACPI_NO_ERROR_MESSAGES)

/******************************************************************************
```



```
 *
 * FUNCTION:    AcpiUtPredefinedWarning
 *
 * PARAMETERS:  ModuleName      - Caller's module name (for error output)
 *              LineNumber      - Caller's line number (for error output)
 *              Pathname        - Full pathname to the node
 *              NodeFlags       - From Namespace node for the method/object
 *              Format          - Printf format string + additional args
 *
 * RETURN:      None
 *
 * DESCRIPTION: Warnings for the predefined validation module. Messages are
 *              only emitted the first time a problem with a particular
 *              method/object is detected. This prevents a flood of error
 *              messages for methods that are repeatedly evaluated.
 *
 *****************************************************************************/

void ACPI_INTERNAL_VAR_XFACE
AcpiUtPredefinedWarning (
    const char              *ModuleName,
    UINT32                  LineNumber,
    char                    *Pathname,
    UINT8                   NodeFlags,
    const char              *Format,
    ...)
{
    va_list                 ArgList;


    /*
     * Warning messages for this method/object will be disabled after the
     * first time a validation fails or an object is successfully repaired.
     */
    if (NodeFlags & ANOBJ_EVALUATED)
    {
        return;
    }

    AcpiOsPrintf (ACPI_MSG_WARNING "%s: ", Pathname);

    va_start (ArgList, Format);
    AcpiOsVprintf (Format, ArgList);
    ACPI_MSG_SUFFIX;
    va_end (ArgList);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtPredefinedInfo
 *
 * PARAMETERS:  ModuleName      - Caller's module name (for error output)
 *              LineNumber      - Caller's line number (for error output)
 *              Pathname        - Full pathname to the node
 *              NodeFlags       - From Namespace node for the method/object
 *              Format          - Printf format string + additional args
 *
 * RETURN:      None
 *
 * DESCRIPTION: Info messages for the predefined validation module. Messages
 *              are only emitted the first time a problem with a particular
 *              method/object is detected. This prevents a flood of
 *              messages for methods that are repeatedly evaluated.
 *
 *****************************************************************************/
```

```
129 void ACPI_INTERNAL_VAR_XFACE
130 AcpiUtPredefinedInfo (
131     const char              *ModuleName,
132     UINT32                  LineNumber,
133     char                    *Pathname,
134     UINT8                   NodeFlags,
135     const char              *Format,
136     ...)
137 {
138     va_list                 ArgList;


141     /*
142      * Warning messages for this method/object will be disabled after the
143      * first time a validation fails or an object is successfully repaired.
144      */
145     if (NodeFlags & ANOBJ_EVALUATED)
146     {
147         return;
148     }

150     AcpiOsPrintf (ACPI_MSG_INFO "%s: ", Pathname);

152     va_start (ArgList, Format);
153     AcpiOsVprintf (Format, ArgList);
154     ACPI_MSG_SUFFIX;
155     va_end (ArgList);
156 }


159 /******************************************************************************
160  *
161  * FUNCTION:    AcpiUtPredefinedBiosError
162  *
163  * PARAMETERS:  ModuleName      - Caller's module name (for error output)
164  *              LineNumber      - Caller's line number (for error output)
165  *              Pathname        - Full pathname to the node
166  *              NodeFlags       - From Namespace node for the method/object
167  *              Format          - Printf format string + additional args
168  *
169  * RETURN:      None
170  *
171  * DESCRIPTION: BIOS error message for predefined names. Messages
172  *              are only emitted the first time a problem with a particular
173  *              method/object is detected. This prevents a flood of
174  *              messages for methods that are repeatedly evaluated.
175  *
176  *****************************************************************************/

178 void ACPI_INTERNAL_VAR_XFACE
179 AcpiUtPredefinedBiosError (
180     const char              *ModuleName,
181     UINT32                  LineNumber,
182     char                    *Pathname,
183     UINT8                   NodeFlags,
184     const char              *Format,
185     ...)
186 {
187     va_list                 ArgList;


190     /*
191      * Warning messages for this method/object will be disabled after the
192      * first time a validation fails or an object is successfully repaired.
193      */
```



```
194     if (NodeFlags & ANOBJ_EVALUATED)
195     {
196         return;
197     }

199     AcpiOsPrintf (ACPI_MSG_BIOS_ERROR "%s: ", Pathname);

201     va_start (ArgList, Format);
202     AcpiOsVprintf (Format, ArgList);
203     ACPI_MSG_SUFFIX;
204     va_end (ArgList);
205 }


208 /******************************************************************************
209  *
210  * FUNCTION:    AcpiUtNamespaceError
211  *
212  * PARAMETERS:  ModuleName          - Caller's module name (for error output)
213  *              LineNumber          - Caller's line number (for error output)
214  *              InternalName        - Name or path of the namespace node
215  *              LookupStatus        - Exception code from NS lookup
216  *
217  * RETURN:      None
218  *
219  * DESCRIPTION: Print error message with the full pathname for the NS node.
220  *
221  *****************************************************************************/

223 void
224 AcpiUtNamespaceError (
225     const char              *ModuleName,
226     UINT32                  LineNumber,
227     const char              *InternalName,
228     ACPI_STATUS             LookupStatus)
229 {
230     ACPI_STATUS             Status;
231     UINT32                  BadName;
232     char                    *Name = NULL;


235     ACPI_MSG_REDIRECT_BEGIN;
236     AcpiOsPrintf (ACPI_MSG_ERROR);

238     if (LookupStatus == AE_BAD_CHARACTER)
239     {
240         /* There is a non-ascii character in the name */

242         ACPI_MOVE_32_TO_32 (&BadName, ACPI_CAST_PTR (UINT32, InternalName));
243         AcpiOsPrintf ("[0x%.8X] (NON-ASCII)", BadName);
244     }
245     else
246     {
247         /* Convert path to external format */

249         Status = AcpiNsExternalizeName (ACPI_UINT32_MAX,
250                     InternalName, NULL, &Name);

252         /* Print target name */

254         if (ACPI_SUCCESS (Status))
255         {
256             AcpiOsPrintf ("[%s]", Name);
257         }
258         else
259         {
```

```
260             AcpiOsPrintf ("[COULD NOT EXTERNALIZE NAME]");
261         }

263         if (Name)
264         {
265             ACPI_FREE (Name);
266         }
267     }

269     AcpiOsPrintf (" Namespace lookup failure, %s",
270         AcpiFormatException (LookupStatus));

272     ACPI_MSG_SUFFIX;
273     ACPI_MSG_REDIRECT_END;
274 }


277 /*******************************************************************************
278  *
279  * FUNCTION:    AcpiUtMethodError
280  *
281  * PARAMETERS:  ModuleName          - Caller's module name (for error output)
282  *              LineNumber          - Caller's line number (for error output)
283  *              Message             - Error message to use on failure
284  *              PrefixNode          - Prefix relative to the path
285  *              Path                - Path to the node (optional)
286  *              MethodStatus        - Execution status
287  *
288  * RETURN:      None
289  *
290  * DESCRIPTION: Print error message with the full pathname for the method.
291  *
292  ******************************************************************************/

294 void
295 AcpiUtMethodError (
296     const char              *ModuleName,
297     UINT32                  LineNumber,
298     const char              *Message,
299     ACPI_NAMESPACE_NODE     *PrefixNode,
300     const char              *Path,
301     ACPI_STATUS             MethodStatus)
302 {
303     ACPI_STATUS             Status;
304     ACPI_NAMESPACE_NODE     *Node = PrefixNode;


307     ACPI_MSG_REDIRECT_BEGIN;
308     AcpiOsPrintf (ACPI_MSG_ERROR);

310     if (Path)
311     {
312         Status = AcpiNsGetNode (PrefixNode, Path, ACPI_NS_NO_UPSEARCH,
313                     &Node);
314         if (ACPI_FAILURE (Status))
315         {
316             AcpiOsPrintf ("[Could not get node by pathname]");
317         }
318     }

320     AcpiNsPrintNodePathname (Node, Message);
321     AcpiOsPrintf (", %s", AcpiFormatException (MethodStatus));

323     ACPI_MSG_SUFFIX;
324     ACPI_MSG_REDIRECT_END;
325 }
```



```
327 #endif /* ACPI_NO_ERROR_MESSAGES */
```

```
**********************************************************
   11359 Thu Dec 26 13:49:38 2013
new/usr/src/common/acpica/components/utilities/uteval.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: uteval - Object evaluation
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTEVAL_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acnamesp.h"


  51 #define _COMPONENT          ACPI_UTILITIES
  52         ACPI_MODULE_NAME    ("uteval")


  55 /*******************************************************************************
  56  *
  57  * FUNCTION:    AcpiUtEvaluateObject
  58  *
  59  * PARAMETERS:  PrefixNode          - Starting node
```



```
  60  *              Path                - Path to object from starting node
  61  *              ExpectedReturnTypes - Bitmap of allowed return types
  62  *              ReturnDesc          - Where a return value is stored
  63  *
  64  * RETURN:      Status
  65  *
  66  * DESCRIPTION: Evaluates a namespace object and verifies the type of the
  67  *              return object. Common code that simplifies accessing objects
  68  *              that have required return objects of fixed types.
  69  *
  70  *              NOTE: Internal function, no parameter validation
  71  *
  72  *****************************************************************************/

  74 ACPI_STATUS
  75 AcpiUtEvaluateObject (
  76     ACPI_NAMESPACE_NODE     *PrefixNode,
  77     char                    *Path,
  78     UINT32                  ExpectedReturnBtypes,
  79     ACPI_OPERAND_OBJECT     **ReturnDesc)
  80 {
  81     ACPI_EVALUATE_INFO      *Info;
  82     ACPI_STATUS             Status;
  83     UINT32                  ReturnBtype;


  86     ACPI_FUNCTION_TRACE (UtEvaluateObject);


  89     /* Allocate the evaluation information block */

  91     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
  92     if (!Info)
  93     {
  94         return_ACPI_STATUS (AE_NO_MEMORY);
  95     }

  97     Info->PrefixNode = PrefixNode;
  98     Info->RelativePathname = Path;
  98     Info->Pathname = Path;

 100     /* Evaluate the object/method */

 102     Status = AcpiNsEvaluate (Info);
 103     if (ACPI_FAILURE (Status))
 104     {
 105         if (Status == AE_NOT_FOUND)
 106         {
 107             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "[%4.4s.%s] was not found\n",
 108                 AcpiUtGetNodeName (PrefixNode), Path));
 109         }
 110         else
 111         {
 112             ACPI_ERROR_METHOD ("Method execution failed",
 113                 PrefixNode, Path, Status);
 114         }

 116         goto Cleanup;
 117     }

 119     /* Did we get a return object? */

 121     if (!Info->ReturnObject)
 122     {
 123         if (ExpectedReturnBtypes)
 124         {
```

```
125             ACPI_ERROR_METHOD ("No object was returned from",
126                 PrefixNode, Path, AE_NOT_EXIST);
127
128             Status = AE_NOT_EXIST;
129         }
130
131         goto Cleanup;
132     }
133
134     /* Map the return object type to the bitmapped type */
135
136     switch ((Info->ReturnObject)->Common.Type)
137     {
138     case ACPI_TYPE_INTEGER:
139
140         ReturnBtype = ACPI_BTYPE_INTEGER;
141         break;
142
143     case ACPI_TYPE_BUFFER:
144
145         ReturnBtype = ACPI_BTYPE_BUFFER;
146         break;
147
148     case ACPI_TYPE_STRING:
149
150         ReturnBtype = ACPI_BTYPE_STRING;
151         break;
152
153     case ACPI_TYPE_PACKAGE:
154
155         ReturnBtype = ACPI_BTYPE_PACKAGE;
156         break;
157
158     default:
159
160         ReturnBtype = 0;
161         break;
162     }
163
164     if ((AcpiGbl_EnableInterpreterSlack) &&
165         (!ExpectedReturnBtypes))
166     {
167         /*
168          * We received a return object, but one was not expected. This can
169          * happen frequently if the "implicit return" feature is enabled.
170          * Just delete the return object and return AE_OK.
171          */
172         AcpiUtRemoveReference (Info->ReturnObject);
173         goto Cleanup;
174     }
175
176     /* Is the return object one of the expected types? */
177
178     if (!(ExpectedReturnBtypes & ReturnBtype))
179     {
180         ACPI_ERROR_METHOD ("Return object type is incorrect",
181             PrefixNode, Path, AE_TYPE);
182
183         ACPI_ERROR ((AE_INFO,
184             "Type returned from %s was incorrect: %s, expected Btypes: 0x%X",
185             Path, AcpiUtGetObjectTypeName (Info->ReturnObject),
186             ExpectedReturnBtypes));
187
188         /* On error exit, we must delete the return object */
189
190         AcpiUtRemoveReference (Info->ReturnObject);
```



```
191         Status = AE_TYPE;
192         goto Cleanup;
193     }
194
195     /* Object type is OK, return it */
196
197     *ReturnDesc = Info->ReturnObject;
198
199 Cleanup:
200     ACPI_FREE (Info);
201     return_ACPI_STATUS (Status);
202 }
______unchanged_portion_omitted_


253 /*******************************************************************************
254  *
255  * FUNCTION:    AcpiUtExecute_STA
256  *
257  * PARAMETERS:  DeviceNode          - Node for the device
258  *              Flags               - Where the status flags are returned
259  *
260  * RETURN:      Status
261  *
262  * DESCRIPTION: Executes _STA for selected device and stores results in
263  *              *Flags. If _STA does not exist, then the device is assumed
264  *              to be present/functional/enabled (as per the ACPI spec).
258  *              *Flags.
265  *
266  *              NOTE: Internal function, no parameter validation
267  *
268  ******************************************************************************/
269
270 ACPI_STATUS
271 AcpiUtExecute_STA (
272     ACPI_NAMESPACE_NODE     *DeviceNode,
273     UINT32                  *Flags)
274 {
275     ACPI_OPERAND_OBJECT     *ObjDesc;
276     ACPI_STATUS             Status;
277
278
279     ACPI_FUNCTION_TRACE (UtExecute_STA);
280
281
282     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__STA,
283                 ACPI_BTYPE_INTEGER, &ObjDesc);
284     if (ACPI_FAILURE (Status))
285     {
286         if (AE_NOT_FOUND == Status)
287         {
288             /*
289              * if _STA does not exist, then (as per the ACPI specification),
290              * the returned flags will indicate that the device is present,
291              * functional, and enabled.
292              */
293             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
294                 "_STA on %4.4s was not found, assuming device is present\n",
295                 AcpiUtGetNodeName (DeviceNode)));
296
297             *Flags = ACPI_UINT32_MAX;
298             Status = AE_OK;
299         }
300
301         return_ACPI_STATUS (Status);
302     }
```

```
 304     /* Extract the status flags */

 306     *Flags = (UINT32) ObjDesc->Integer.Value;

 308     /* On exit, we must delete the return object */

 310     AcpiUtRemoveReference (ObjDesc);
 311     return_ACPI_STATUS (Status);
 312 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    5238 Thu Dec 26 13:49:39 2013
new/usr/src/common/acpica/components/utilities/utexcep.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utexcep - Exception code support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTEXCEP_C__
#define EXPORT_ACPI_INTERFACES

#define ACPI_DEFINE_EXCEPTION_TABLE
#include "acpi.h"
#include "accommon.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utexcep")


/******************************************************************************
 *
 * FUNCTION:    AcpiFormatException
 *
```



```
 * PARAMETERS:  Status              - The ACPI_STATUS code to be formatted
 *
 * RETURN:      A string containing the exception text. A valid pointer is
 *              always returned.
 *
 * DESCRIPTION: This function translates an ACPI exception into an ASCII
 *              string. Returns "unknown status" string for invalid codes.
 *
 *****************************************************************************/

const char *
AcpiFormatException (
    ACPI_STATUS             Status)
{
    const ACPI_EXCEPTION_INFO   *Exception;


    ACPI_FUNCTION_ENTRY ();


    Exception = AcpiUtValidateException (Status);
    if (!Exception)
    {
        /* Exception code was not recognized */

        ACPI_ERROR ((AE_INFO,
            "Unknown exception code: 0x%8.8X", Status));

        return ("UNKNOWN_STATUS_CODE");
    }

    return (Exception->Name);
}

ACPI_EXPORT_SYMBOL (AcpiFormatException)


/******************************************************************************
 *
 * FUNCTION:    AcpiUtValidateException
 *
 * PARAMETERS:  Status              - The ACPI_STATUS code to be formatted
 *
 * RETURN:      A string containing the exception text. NULL if exception is
 *              not valid.
 *
 * DESCRIPTION: This function validates and translates an ACPI exception into
 *              an ASCII string.
 *
 *****************************************************************************/

const ACPI_EXCEPTION_INFO *
AcpiUtValidateException (
    ACPI_STATUS             Status)
{
    UINT32                  SubStatus;
    const ACPI_EXCEPTION_INFO   *Exception = NULL;


    ACPI_FUNCTION_ENTRY ();


    /*
     * Status is composed of two parts, a "type" and an actual code
     */
    SubStatus = (Status & ~AE_CODE_MASK);
```

```
128     switch (Status & AE_CODE_MASK)
129     {
130     case AE_CODE_ENVIRONMENTAL:

132         if (SubStatus <= AE_CODE_ENV_MAX)
133         {
134             Exception = &AcpiGbl_ExceptionNames_Env [SubStatus];
135         }
136         break;

138     case AE_CODE_PROGRAMMER:

140         if (SubStatus <= AE_CODE_PGM_MAX)
141         {
142             Exception = &AcpiGbl_ExceptionNames_Pgm [SubStatus];
143         }
144         break;

146     case AE_CODE_ACPI_TABLES:

148         if (SubStatus <= AE_CODE_TBL_MAX)
149         {
150             Exception = &AcpiGbl_ExceptionNames_Tbl [SubStatus];
151         }
152         break;

154     case AE_CODE_AML:

156         if (SubStatus <= AE_CODE_AML_MAX)
157         {
158             Exception = &AcpiGbl_ExceptionNames_Aml [SubStatus];
159         }
160         break;

162     case AE_CODE_CONTROL:

164         if (SubStatus <= AE_CODE_CTRL_MAX)
165         {
166             Exception = &AcpiGbl_ExceptionNames_Ctrl [SubStatus];
167         }
168         break;

170     default:

172         break;
173     }

175     if (!Exception || !Exception->Name)
176     {
177         return (NULL);
178     }

180     return (Exception);
181 }
```

```
**********************************************************
   14312 Thu Dec 26 13:49:39 2013
new/usr/src/common/acpica/components/utilities/utglobal.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utglobal - Global variables for the ACPI subsystem
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTGLOBAL_C__
  45 #define EXPORT_ACPI_INTERFACES
  46 #define DEFINE_ACPI_GLOBALS

  48 #include "acpi.h"
  49 #include "accommon.h"

  51 #define _COMPONENT          ACPI_UTILITIES
  52         ACPI_MODULE_NAME    ("utglobal")


  55 /******************************************************************************
  56  *
  57  * Static global variable initialization.
  58  *
```



```
  59  *****************************************************************************/

  61 /*
  62  * We want the debug switches statically initialized so they
  63  * are already set when the debugger is entered.
  64  */

  66 /* Debug switch - level and trace mask */

  68 #ifdef ACPI_DEBUG_OUTPUT
  69 UINT32                      AcpiDbgLevel = ACPI_DEBUG_DEFAULT;
  70 #else
  71 UINT32                      AcpiDbgLevel = ACPI_NORMAL_DEFAULT;
  72 #endif

  74 /* Debug switch - layer (component) mask */

  76 UINT32                      AcpiDbgLayer = ACPI_COMPONENT_DEFAULT;
  77 UINT32                      AcpiGbl_NestingLevel = 0;

  79 /* Debugger globals */

  81 BOOLEAN                     AcpiGbl_DbTerminateThreads = FALSE;
  82 BOOLEAN                     AcpiGbl_AbortMethod = FALSE;
  83 BOOLEAN                     AcpiGbl_MethodExecuting = FALSE;

  85 /* System flags */

  87 UINT32                      AcpiGbl_StartupFlags = 0;

  89 /* System starts uninitialized */

  91 BOOLEAN                     AcpiGbl_Shutdown = TRUE;

  93 const char                  *AcpiGbl_SleepStateNames[ACPI_S_STATE_COUNT] =
  94 {
  95     "\\_S0_",
  96     "\\_S1_",
  97     "\\_S2_",
  98     "\\_S3_",
  99     "\\_S4_",
 100     "\\_S5_"
 101 };
______unchanged_portion_omitted_


 159 #if (!ACPI_REDUCED_HARDWARE)
 160 /******************************************************************************
 161  *
 162  * Event and Hardware globals
 163  *
 164  *****************************************************************************/

 166 ACPI_BIT_REGISTER_INFO      AcpiGbl_BitRegisterInfo[ACPI_NUM_BITREG] =
 167 {
 168     /* Name                                     Parent Register             Regi

 170     /* ACPI_BITREG_TIMER_STATUS         */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 171     /* ACPI_BITREG_BUS_MASTER_STATUS    */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 172     /* ACPI_BITREG_GLOBAL_LOCK_STATUS   */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 173     /* ACPI_BITREG_POWER_BUTTON_STATUS  */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 174     /* ACPI_BITREG_SLEEP_BUTTON_STATUS  */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 175     /* ACPI_BITREG_RT_CLOCK_STATUS      */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 176     /* ACPI_BITREG_WAKE_STATUS          */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
 177     /* ACPI_BITREG_PCIEXP_WAKE_STATUS   */   {ACPI_REGISTER_PM1_STATUS,    ACPI_B
```

```
179     /* ACPI_BITREG_TIMER_ENABLE         */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B
180     /* ACPI_BITREG_GLOBAL_LOCK_ENABLE   */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B
181     /* ACPI_BITREG_POWER_BUTTON_ENABLE  */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B
182     /* ACPI_BITREG_SLEEP_BUTTON_ENABLE  */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B
183     /* ACPI_BITREG_RT_CLOCK_ENABLE      */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B
184     /* ACPI_BITREG_PCIEXP_WAKE_DISABLE  */   {ACPI_REGISTER_PM1_ENABLE,   ACPI_B

186     /* ACPI_BITREG_SCI_ENABLE           */   {ACPI_REGISTER_PM1_CONTROL,  ACPI_B
187     /* ACPI_BITREG_BUS_MASTER_RLD       */   {ACPI_REGISTER_PM1_CONTROL,  ACPI_B
188     /* ACPI_BITREG_GLOBAL_LOCK_RELEASE  */   {ACPI_REGISTER_PM1_CONTROL,  ACPI_B
189     /* ACPI_BITREG_SLEEP_TYPE           */   {ACPI_REGISTER_PM1_CONTROL,  ACPI_B
190     /* ACPI_BITREG_SLEEP_ENABLE         */   {ACPI_REGISTER_PM1_CONTROL,  ACPI_B

192     /* ACPI_BITREG_ARB_DIS              */   {ACPI_REGISTER_PM2_CONTROL,  ACPI_B
193 };
      unchanged_portion_omitted_
204 #endif /* !ACPI_REDUCED_HARDWARE */


207 /******************************************************************************
208  *
209  * FUNCTION:    AcpiUtInitGlobals
210  *
211  * PARAMETERS:  None
212  *
213  * RETURN:      Status
214  *
215  * DESCRIPTION: Initialize ACPICA globals. All globals that require specific
216  *              initialization should be initialized here. This allows for
217  *              a warm restart.
212  * DESCRIPTION: Init ACPICA globals.  All globals that require specific
213  *              initialization should be initialized here!
218  *
219  *****************************************************************************/

221 ACPI_STATUS
222 AcpiUtInitGlobals (
223     void)
224 {
225     ACPI_STATUS             Status;
226     UINT32                  i;


229     ACPI_FUNCTION_TRACE (UtInitGlobals);


232     /* Create all memory caches */

234     Status = AcpiUtCreateCaches ();
235     if (ACPI_FAILURE (Status))
236     {
237         return_ACPI_STATUS (Status);
238     }

240     /* Address Range lists */

242     for (i = 0; i < ACPI_ADDRESS_RANGE_MAX; i++)
243     {
244         AcpiGbl_AddressRangeList[i] = NULL;
245     }

247     /* Mutex locked flags */

249     for (i = 0; i < ACPI_NUM_MUTEX; i++)
250     {
251         AcpiGbl_MutexInfo[i].Mutex          = NULL;
```



```
252         AcpiGbl_MutexInfo[i].ThreadId       = ACPI_MUTEX_NOT_ACQUIRED;
253         AcpiGbl_MutexInfo[i].UseCount       = 0;
254     }

256     for (i = 0; i < ACPI_NUM_OWNERID_MASKS; i++)
257     {
258         AcpiGbl_OwnerIdMask[i]              = 0;
259     }

261     /* Last OwnerID is never valid */

263     AcpiGbl_OwnerIdMask[ACPI_NUM_OWNERID_MASKS - 1] = 0x80000000;

265     /* Event counters */

267     AcpiMethodCount                     = 0;
268     AcpiSciCount                        = 0;
269     AcpiGpeCount                        = 0;

271     for (i = 0; i < ACPI_NUM_FIXED_EVENTS; i++)
272     {
273         AcpiFixedEventCount[i]              = 0;
274     }

276 #if (!ACPI_REDUCED_HARDWARE)
265     /* GPE support */

278     /* GPE/SCI support */

280     AcpiGbl_AllGpesInitialized          = FALSE;
281     AcpiGbl_GpeXruptListHead            = NULL;
282     AcpiGbl_GpeFadtBlocks[0]            = NULL;
283     AcpiGbl_GpeFadtBlocks[1]            = NULL;
284     AcpiCurrentGpeCount                 = 0;

286     AcpiGbl_GlobalEventHandler          = NULL;
287     AcpiGbl_SciHandlerList              = NULL;

289 #endif /* !ACPI_REDUCED_HARDWARE */

291     /* Global handlers */

293     AcpiGbl_GlobalNotify[0].Handler     = NULL;
294     AcpiGbl_GlobalNotify[1].Handler     = NULL;
275     AcpiGbl_SystemNotify.Handler        = NULL;
276     AcpiGbl_DeviceNotify.Handler        = NULL;
295     AcpiGbl_ExceptionHandler            = NULL;
296     AcpiGbl_InitHandler                 = NULL;
297     AcpiGbl_TableHandler                = NULL;
298     AcpiGbl_InterfaceHandler            = NULL;
281     AcpiGbl_GlobalEventHandler          = NULL;

300     /* Global Lock support */

302     AcpiGbl_GlobalLockSemaphore         = NULL;
303     AcpiGbl_GlobalLockMutex             = NULL;
304     AcpiGbl_GlobalLockAcquired          = FALSE;
305     AcpiGbl_GlobalLockHandle            = 0;
306     AcpiGbl_GlobalLockPresent           = FALSE;

308     /* Miscellaneous variables */

310     AcpiGbl_DSDT                        = NULL;
311     AcpiGbl_CmSingleStep                = FALSE;
312     AcpiGbl_DbTerminateThreads          = FALSE;
313     AcpiGbl_Shutdown                    = FALSE;
```

```
 314     AcpiGbl_NsLookupCount               = 0;
 315     AcpiGbl_PsFindCount                 = 0;
 316     AcpiGbl_AcpiHardwarePresent         = TRUE;
 317     AcpiGbl_LastOwnerIdIndex            = 0;
 318     AcpiGbl_NextOwnerIdOffset           = 0;
 319     AcpiGbl_TraceMethodName             = 0;
 320     AcpiGbl_TraceDbgLevel               = 0;
 321     AcpiGbl_TraceDbgLayer               = 0;
 322     AcpiGbl_DebuggerConfiguration       = DEBUGGER_THREADING;
 323     AcpiGbl_DbOutputFlags               = ACPI_DB_CONSOLE_OUTPUT;
 307     AcpiGbl_OsiData                     = 0;
 324     AcpiGbl_OsiMutex                    = NULL;
 325     AcpiGbl_RegMethodsExecuted          = FALSE;

 327     /* Hardware oriented */

 329     AcpiGbl_EventsInitialized           = FALSE;
 330     AcpiGbl_SystemAwakeAndRunning       = TRUE;

 332     /* Namespace */

 334     AcpiGbl_ModuleCodeList              = NULL;
 335     AcpiGbl_RootNode                    = NULL;
 336     AcpiGbl_RootNodeStruct.Name.Integer = ACPI_ROOT_NAME;
 337     AcpiGbl_RootNodeStruct.DescriptorType = ACPI_DESC_TYPE_NAMED;
 338     AcpiGbl_RootNodeStruct.Type         = ACPI_TYPE_DEVICE;
 339     AcpiGbl_RootNodeStruct.Parent       = NULL;
 340     AcpiGbl_RootNodeStruct.Child        = NULL;
 341     AcpiGbl_RootNodeStruct.Peer         = NULL;
 342     AcpiGbl_RootNodeStruct.Object       = NULL;


 345 #ifdef ACPI_DISASSEMBLER
 346     AcpiGbl_ExternalList                = NULL;
 347     AcpiGbl_NumExternalMethods          = 0;
 348     AcpiGbl_ResolvedExternalMethods     = 0;
 349 #endif

 351 #ifdef ACPI_DEBUG_OUTPUT
 352     AcpiGbl_LowestStackPointer          = ACPI_CAST_PTR (ACPI_SIZE, ACPI_SIZE_MA
 353 #endif

 355 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
 356     AcpiGbl_DisplayFinalMemStats        = FALSE;
 357     AcpiGbl_DisableMemTracking          = FALSE;
 358 #endif

 360     return_ACPI_STATUS (AE_OK);
 361 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   13188 Thu Dec 26 13:49:39 2013
new/usr/src/common/acpica/components/utilities/utids.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utids - support for device IDs - HID, UID, CID
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTIDS_C__

#include "acpi.h"
#include "accommon.h"
#include "acinterp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utids")


/******************************************************************************
 *
 * FUNCTION:    AcpiUtExecute_HID
 *
 * PARAMETERS:  DeviceNode          - Node for the device
```



```
 *              ReturnId            - Where the string HID is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Executes the _HID control method that returns the hardware
 *              ID of the device. The HID is either an 32-bit encoded EISAID
 *              Integer or a String. A string is always returned. An EISAID
 *              is converted to a string.
 *
 *              NOTE: Internal function, no parameter validation
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtExecute_HID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
    ACPI_DEVICE_ID          **ReturnId)
{
    ACPI_OPERAND_OBJECT     *ObjDesc;
    ACPI_PNP_DEVICE_ID      *Hid;
    ACPI_DEVICE_ID          *Hid;
    UINT32                  Length;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (UtExecute_HID);


    Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__HID,
                ACPI_BTYPE_INTEGER | ACPI_BTYPE_STRING, &ObjDesc);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Get the size of the String to be returned, includes null terminator */

    if (ObjDesc->Common.Type == ACPI_TYPE_INTEGER)
    {
        Length = ACPI_EISAID_STRING_SIZE;
    }
    else
    {
        Length = ObjDesc->String.Length + 1;
    }

    /* Allocate a buffer for the HID */

    Hid = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_PNP_DEVICE_ID) + (ACPI_SIZE) Length
    Hid = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_DEVICE_ID) + (ACPI_SIZE) Length);
    if (!Hid)
    {
        Status = AE_NO_MEMORY;
        goto Cleanup;
    }

    /* Area for the string starts after PNP_DEVICE_ID struct */
    /* Area for the string starts after DEVICE_ID struct */

    Hid->String = ACPI_ADD_PTR (char, Hid, sizeof (ACPI_PNP_DEVICE_ID));
    Hid->String = ACPI_ADD_PTR (char, Hid, sizeof (ACPI_DEVICE_ID));

    /* Convert EISAID to a string or simply copy existing string */

    if (ObjDesc->Common.Type == ACPI_TYPE_INTEGER)
```

```
121     {
122         AcpiExEisaIdToString (Hid->String, ObjDesc->Integer.Value);
123     }
124     else
125     {
126         ACPI_STRCPY (Hid->String, ObjDesc->String.Pointer);
127     }

129     Hid->Length = Length;
130     *ReturnId = Hid;


133 Cleanup:

135     /* On exit, we must delete the return object */

137     AcpiUtRemoveReference (ObjDesc);
138     return_ACPI_STATUS (Status);
139 }


142 /******************************************************************************
143  *
144  * FUNCTION:    AcpiUtExecute_SUB
145  *
146  * PARAMETERS:  DeviceNode          - Node for the device
147  *              ReturnId            - Where the _SUB is returned
148  *
149  * RETURN:      Status
150  *
151  * DESCRIPTION: Executes the _SUB control method that returns the subsystem
152  *              ID of the device. The _SUB value is always a string containing
153  *              either a valid PNP or ACPI ID.
154  *
155  *              NOTE: Internal function, no parameter validation
156  *
157  *****************************************************************************/

159 ACPI_STATUS
160 AcpiUtExecute_SUB (
161     ACPI_NAMESPACE_NODE     *DeviceNode,
162     ACPI_PNP_DEVICE_ID      **ReturnId)
163 {
164     ACPI_OPERAND_OBJECT     *ObjDesc;
165     ACPI_PNP_DEVICE_ID      *Sub;
166     UINT32                  Length;
167     ACPI_STATUS             Status;


170     ACPI_FUNCTION_TRACE (UtExecute_SUB);


173     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__SUB,
174                 ACPI_BTYPE_STRING, &ObjDesc);
175     if (ACPI_FAILURE (Status))
176     {
177         return_ACPI_STATUS (Status);
178     }

180     /* Get the size of the String to be returned, includes null terminator */

182     Length = ObjDesc->String.Length + 1;

184     /* Allocate a buffer for the SUB */

186     Sub = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_PNP_DEVICE_ID) + (ACPI_SIZE) Length
```



```
187     if (!Sub)
188     {
189         Status = AE_NO_MEMORY;
190         goto Cleanup;
191     }

193     /* Area for the string starts after PNP_DEVICE_ID struct */

195     Sub->String = ACPI_ADD_PTR (char, Sub, sizeof (ACPI_PNP_DEVICE_ID));

197     /* Simply copy existing string */

199     ACPI_STRCPY (Sub->String, ObjDesc->String.Pointer);
200     Sub->Length = Length;
201     *ReturnId = Sub;


204 Cleanup:

206     /* On exit, we must delete the return object */

208     AcpiUtRemoveReference (ObjDesc);
209     return_ACPI_STATUS (Status);
210 }


213 /******************************************************************************
214  *
215  * FUNCTION:    AcpiUtExecute_UID
216  *
217  * PARAMETERS:  DeviceNode          - Node for the device
218  *              ReturnId            - Where the string UID is returned
219  *
220  * RETURN:      Status
221  *
222  * DESCRIPTION: Executes the _UID control method that returns the unique
223  *              ID of the device. The UID is either a 64-bit Integer (NOT an
224  *              EISAID) or a string. Always returns a string. A 64-bit integer
225  *              is converted to a decimal string.
226  *
227  *              NOTE: Internal function, no parameter validation
228  *
229  *****************************************************************************/

231 ACPI_STATUS
232 AcpiUtExecute_UID (
233     ACPI_NAMESPACE_NODE     *DeviceNode,
234     ACPI_PNP_DEVICE_ID      **ReturnId)
163     ACPI_DEVICE_ID          **ReturnId)
235 {
236     ACPI_OPERAND_OBJECT     *ObjDesc;
237     ACPI_PNP_DEVICE_ID      *Uid;
166     ACPI_DEVICE_ID          *Uid;
238     UINT32                  Length;
239     ACPI_STATUS             Status;


242     ACPI_FUNCTION_TRACE (UtExecute_UID);


245     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__UID,
246                 ACPI_BTYPE_INTEGER | ACPI_BTYPE_STRING, &ObjDesc);
247     if (ACPI_FAILURE (Status))
248     {
249         return_ACPI_STATUS (Status);
250     }
```

```
252     /* Get the size of the String to be returned, includes null terminator */

254     if (ObjDesc->Common.Type == ACPI_TYPE_INTEGER)
255     {
256         Length = ACPI_MAX64_DECIMAL_DIGITS + 1;
257     }
258     else
259     {
260         Length = ObjDesc->String.Length + 1;
261     }

263     /* Allocate a buffer for the UID */

265     Uid = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_PNP_DEVICE_ID) + (ACPI_SIZE) Length
194     Uid = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_DEVICE_ID) + (ACPI_SIZE) Length);
266     if (!Uid)
267     {
268         Status = AE_NO_MEMORY;
269         goto Cleanup;
270     }

272     /* Area for the string starts after PNP_DEVICE_ID struct */
201     /* Area for the string starts after DEVICE_ID struct */

274     Uid->String = ACPI_ADD_PTR (char, Uid, sizeof (ACPI_PNP_DEVICE_ID));
203     Uid->String = ACPI_ADD_PTR (char, Uid, sizeof (ACPI_DEVICE_ID));

276     /* Convert an Integer to string, or just copy an existing string */

278     if (ObjDesc->Common.Type == ACPI_TYPE_INTEGER)
279     {
280         AcpiExIntegerToString (Uid->String, ObjDesc->Integer.Value);
281     }
282     else
283     {
284         ACPI_STRCPY (Uid->String, ObjDesc->String.Pointer);
285     }

287     Uid->Length = Length;
288     *ReturnId = Uid;


291 Cleanup:

293     /* On exit, we must delete the return object */

295     AcpiUtRemoveReference (ObjDesc);
296     return_ACPI_STATUS (Status);
297 }


300 /******************************************************************************
301  *
302  * FUNCTION:    AcpiUtExecute_CID
303  *
304  * PARAMETERS:  DeviceNode          - Node for the device
305  *              ReturnCidList       - Where the CID list is returned
306  *
307  * RETURN:      Status, list of CID strings
308  *
309  * DESCRIPTION: Executes the _CID control method that returns one or more
310  *              compatible hardware IDs for the device.
311  *
312  *              NOTE: Internal function, no parameter validation
313  *
```



```
314  * A _CID method can return either a single compatible ID or a package of
315  * compatible IDs. Each compatible ID can be one of the following:
316  * 1) Integer (32 bit compressed EISA ID) or
317  * 2) String (PCI ID format, e.g. "PCI\VEN_vvvv&DEV_dddd&SUBSYS_ssssssss")
318  *
319  * The Integer CIDs are converted to string format by this function.
320  *
321  *****************************************************************************/

323 ACPI_STATUS
324 AcpiUtExecute_CID (
325     ACPI_NAMESPACE_NODE     *DeviceNode,
326     ACPI_PNP_DEVICE_ID_LIST **ReturnCidList)
255     ACPI_DEVICE_ID_LIST     **ReturnCidList)
327 {
328     ACPI_OPERAND_OBJECT     **CidObjects;
329     ACPI_OPERAND_OBJECT     *ObjDesc;
330     ACPI_PNP_DEVICE_ID_LIST *CidList;
259     ACPI_DEVICE_ID_LIST     *CidList;
331     char                    *NextIdString;
332     UINT32                  StringAreaSize;
333     UINT32                  Length;
334     UINT32                  CidListSize;
335     ACPI_STATUS             Status;
336     UINT32                  Count;
337     UINT32                  i;


340     ACPI_FUNCTION_TRACE (UtExecute_CID);


343     /* Evaluate the _CID method for this device */

345     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__CID,
346                 ACPI_BTYPE_INTEGER | ACPI_BTYPE_STRING | ACPI_BTYPE_PACKAGE,
347                 &ObjDesc);
348     if (ACPI_FAILURE (Status))
349     {
350         return_ACPI_STATUS (Status);
351     }

353     /*
354      * Get the count and size of the returned _CIDs. _CID can return either
355      * a Package of Integers/Strings or a single Integer or String.
356      * Note: This section also validates that all CID elements are of the
357      * correct type (Integer or String).
358      */
359     if (ObjDesc->Common.Type == ACPI_TYPE_PACKAGE)
360     {
361         Count = ObjDesc->Package.Count;
362         CidObjects = ObjDesc->Package.Elements;
363     }
364     else /* Single Integer or String CID */
365     {
366         Count = 1;
367         CidObjects = &ObjDesc;
368     }

370     StringAreaSize = 0;
371     for (i = 0; i < Count; i++)
372     {
373         /* String lengths include null terminator */

375         switch (CidObjects[i]->Common.Type)
376         {
377         case ACPI_TYPE_INTEGER:
```

```
379             StringAreaSize += ACPI_EISAID_STRING_SIZE;
380             break;

382         case ACPI_TYPE_STRING:

384             StringAreaSize += CidObjects[i]->String.Length + 1;
385             break;

387         default:

389             Status = AE_TYPE;
390             goto Cleanup;
391         }
392     }

394     /*
395      * Now that we know the length of the CIDs, allocate return buffer:
396      * 1) Size of the base structure +
397      * 2) Size of the CID PNP_DEVICE_ID array +
323      * 2) Size of the CID DEVICE_ID array +
398      * 3) Size of the actual CID strings
399      */
400     CidListSize = sizeof (ACPI_PNP_DEVICE_ID_LIST) +
401         ((Count - 1) * sizeof (ACPI_PNP_DEVICE_ID)) +
326     CidListSize = sizeof (ACPI_DEVICE_ID_LIST) +
327         ((Count - 1) * sizeof (ACPI_DEVICE_ID)) +
402         StringAreaSize;

404     CidList = ACPI_ALLOCATE_ZEROED (CidListSize);
405     if (!CidList)
406     {
407         Status = AE_NO_MEMORY;
408         goto Cleanup;
409     }

411     /* Area for CID strings starts after the CID PNP_DEVICE_ID array */
337     /* Area for CID strings starts after the CID DEVICE_ID array */

413     NextIdString = ACPI_CAST_PTR (char, CidList->Ids) +
414         ((ACPI_SIZE) Count * sizeof (ACPI_PNP_DEVICE_ID));
340         ((ACPI_SIZE) Count * sizeof (ACPI_DEVICE_ID));

416     /* Copy/convert the CIDs to the return buffer */

418     for (i = 0; i < Count; i++)
419     {
420         if (CidObjects[i]->Common.Type == ACPI_TYPE_INTEGER)
421         {
422             /* Convert the Integer (EISAID) CID to a string */

424             AcpiExEisaIdToString (NextIdString, CidObjects[i]->Integer.Value);
425             Length = ACPI_EISAID_STRING_SIZE;
426         }
427         else /* ACPI_TYPE_STRING */
428         {
429             /* Copy the String CID from the returned object */

431             ACPI_STRCPY (NextIdString, CidObjects[i]->String.Pointer);
432             Length = CidObjects[i]->String.Length + 1;
433         }

435         CidList->Ids[i].String = NextIdString;
436         CidList->Ids[i].Length = Length;
437         NextIdString += Length;
438     }
```



```
440     /* Finish the CID list */

442     CidList->Count = Count;
443     CidList->ListSize = CidListSize;
444     *ReturnCidList = CidList;


447 Cleanup:

449     /* On exit, we must delete the _CID return object */

451     AcpiUtRemoveReference (ObjDesc);
452     return_ACPI_STATUS (Status);
453 }
```

```
**********************************************************
    5073 Thu Dec 26 13:49:40 2013
new/usr/src/common/acpica/components/utilities/utinit.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utinit - Common ACPI subsystem initialization
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTINIT_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"
#include "acevents.h"
#include "actables.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utinit")

/* Local prototypes */

static void AcpiUtTerminate (
    void);
```



```
#if (!ACPI_REDUCED_HARDWARE)

static void
AcpiUtFreeGpeLists (
    void);

#else

#define AcpiUtFreeGpeLists()
#endif /* !ACPI_REDUCED_HARDWARE */


#if (!ACPI_REDUCED_HARDWARE)
/******************************************************************************
 *
 * FUNCTION:    AcpiUtFreeGpeLists
 * FUNCTION:    AcpiUtTerminate
 *
 * PARAMETERS:  none
 *
 * RETURN:      none
 *
 * DESCRIPTION: Free global GPE lists
 * DESCRIPTION: Free global memory
 *
 *****************************************************************************/

static void
AcpiUtFreeGpeLists (
AcpiUtTerminate (
    void)
{
    ACPI_GPE_BLOCK_INFO     *GpeBlock;
    ACPI_GPE_BLOCK_INFO     *NextGpeBlock;
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo;
    ACPI_GPE_XRUPT_INFO     *NextGpeXruptInfo;


    ACPI_FUNCTION_TRACE (UtTerminate);


    /* Free global GPE blocks and related info structures */

    GpeXruptInfo = AcpiGbl_GpeXruptListHead;
    while (GpeXruptInfo)
    {
        GpeBlock = GpeXruptInfo->GpeBlockListHead;
        while (GpeBlock)
        {
            NextGpeBlock = GpeBlock->Next;
            ACPI_FREE (GpeBlock->EventInfo);
            ACPI_FREE (GpeBlock->RegisterInfo);
            ACPI_FREE (GpeBlock);

            GpeBlock = NextGpeBlock;
        }
        NextGpeXruptInfo = GpeXruptInfo->Next;
        ACPI_FREE (GpeXruptInfo);
        GpeXruptInfo = NextGpeXruptInfo;
    }
}
#endif /* !ACPI_REDUCED_HARDWARE */


/******************************************************************************
```

```
 120  *
 121  * FUNCTION:    AcpiUtTerminate
 122  *
 123  * PARAMETERS:  none
 124  *
 125  * RETURN:      none
 126  *
 127  * DESCRIPTION: Free global memory
 128  *
 129  ******************************************************************************/

 131 static void
 132 AcpiUtTerminate (
 133     void)
 134 {
 135     ACPI_FUNCTION_TRACE (UtTerminate);

 137     AcpiUtFreeGpeLists ();
 138     AcpiUtDeleteAddressLists ();
 139     return_VOID;
 140 }
_____unchanged_portion_omitted_
```

****
5989 Thu Dec 26 13:49:40 2013
new/usr/src/common/acpica/components/utilities/utlock.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utlock - Reader/Writer lock interfaces
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTLOCK_C__

  46 #include "acpi.h"
  47 #include "accommon.h"


  50 #define _COMPONENT          ACPI_UTILITIES
  51         ACPI_MODULE_NAME    ("utlock")


  54 /******************************************************************************
  55  *
  56  * FUNCTION:    AcpiUtCreateRwLock
  57  *              AcpiUtDeleteRwLock
  58  *
  59  * PARAMETERS:  Lock                - Pointer to a valid RW lock
```



```
  60  *
  61  * RETURN:      Status
  62  *
  63  * DESCRIPTION: Reader/writer lock creation and deletion interfaces.
  64  *
  65  *****************************************************************************/

  67 ACPI_STATUS
  68 AcpiUtCreateRwLock (
  69     ACPI_RW_LOCK            *Lock)
  70 {
  71     ACPI_STATUS             Status;


  74     Lock->NumReaders = 0;
  75     Status = AcpiOsCreateMutex (&Lock->ReaderMutex);
  76     if (ACPI_FAILURE (Status))
  77     {
  78         return (Status);
  79     }

  81     Status = AcpiOsCreateMutex (&Lock->WriterMutex);
  82     return (Status);
  83 }
```

unchanged_portion_omitted_

```
**********************************************************
   11267 Thu Dec 26 13:49:41 2013
new/usr/src/common/acpica/components/utilities/utmath.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utmath - Integer math support routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __UTMATH_C__

  47 #include "acpi.h"
  48 #include "accommon.h"


  51 #define _COMPONENT          ACPI_UTILITIES
  52         ACPI_MODULE_NAME    ("utmath")

  54 /*
  55  * Optional support for 64-bit double-precision integer divide. This code
  56  * is configurable and is implemented in order to support 32-bit kernel
  57  * environments where a 64-bit double-precision math library is not available.
  58  *
  59  * Support for a more normal 64-bit divide/modulo (with check for a divide-
```



```
  60  * by-zero) appears after this optional section of code.
  61  */
  62 #ifndef ACPI_USE_NATIVE_DIVIDE

  64 /* Structures used only for 64-bit divide */

  66 typedef struct uint64_struct
  67 {
  68     UINT32                          Lo;
  69     UINT32                          Hi;

  71 } UINT64_STRUCT;
______unchanged_portion_omitted_

 377 #endif
```

```
**********************************************************
   13127 Thu Dec 26 13:49:41 2013
new/usr/src/common/acpica/components/utilities/utmisc.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utmisc - common utility procedures
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTMISC_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utmisc")


/******************************************************************************
 *
 * FUNCTION:    AcpiUtIsPciRootBridge
 *
 * PARAMETERS:  Id              - The HID/CID in string format
```



```
 *
 * RETURN:      TRUE if the Id is a match for a PCI/PCI-Express Root Bridge
 *
 * DESCRIPTION: Determine if the input ID is a PCI Root Bridge ID.
 *
 *****************************************************************************/

BOOLEAN
AcpiUtIsPciRootBridge (
    char                    *Id)
{

    /*
     * Check if this is a PCI root bridge.
     * ACPI 3.0+: check for a PCI Express root also.
     */
    if (!(ACPI_STRCMP (Id,
            PCI_ROOT_HID_STRING)) ||

        !(ACPI_STRCMP (Id,
            PCI_EXPRESS_ROOT_HID_STRING)))
    {
        return (TRUE);
    }

    return (FALSE);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtIsAmlTable
 *
 * PARAMETERS:  Table               - An ACPI table
 *
 * RETURN:      TRUE if table contains executable AML; FALSE otherwise
 *
 * DESCRIPTION: Check ACPI Signature for a table that contains AML code.
 *              Currently, these are DSDT,SSDT,PSDT. All other table types are
 *              data tables that do not contain AML code.
 *
 *****************************************************************************/

BOOLEAN
AcpiUtIsAmlTable (
    ACPI_TABLE_HEADER       *Table)
{

    /* These are the only tables that contain executable AML */

    if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_DSDT) ||
        ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_PSDT) ||
        ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_SSDT))
    {
        return (TRUE);
    }

    return (FALSE);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtDwordByteSwap
 *
 * PARAMETERS:  Value           - Value to be converted
```

```
 *
 * RETURN:      UINT32 integer with bytes swapped
 *
 * DESCRIPTION: Convert a 32-bit value to big-endian (swap the bytes)
 *
 ******************************************************************************/

UINT32
AcpiUtDwordByteSwap (
    UINT32                  Value)
{
    union
    {
        UINT32              Value;
        UINT8               Bytes[4];
    } Out;
    union
    {
        UINT32              Value;
        UINT8               Bytes[4];
    } In;


    ACPI_FUNCTION_ENTRY ();


    In.Value = Value;

    Out.Bytes[0] = In.Bytes[3];
    Out.Bytes[1] = In.Bytes[2];
    Out.Bytes[2] = In.Bytes[1];
    Out.Bytes[3] = In.Bytes[0];

    return (Out.Value);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtSetIntegerWidth
 *
 * PARAMETERS:  Revision            From DSDT header
 *
 * RETURN:      None
 *
 * DESCRIPTION: Set the global integer bit width based upon the revision
 *              of the DSDT. For Revision 1 and 0, Integers are 32 bits.
 *              For Revision 2 and above, Integers are 64 bits. Yes, this
 *              makes a difference.
 *
 ******************************************************************************/

void
AcpiUtSetIntegerWidth (
    UINT8                   Revision)
{

    if (Revision < 2)
    {
        /* 32-bit case */

        AcpiGbl_IntegerBitWidth    = 32;
        AcpiGbl_IntegerNybbleWidth = 8;
        AcpiGbl_IntegerByteWidth   = 4;
    }
    else
```



```
    {
        /* 64-bit case (ACPI 2.0+) */

        AcpiGbl_IntegerBitWidth    = 64;
        AcpiGbl_IntegerNybbleWidth = 16;
        AcpiGbl_IntegerByteWidth   = 8;
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtCreateUpdateStateAndPush
 *
 * PARAMETERS:  Object          - Object to be added to the new state
 *              Action          - Increment/Decrement
 *              StateList       - List the state will be added to
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create a new state and push it
 *
 ******************************************************************************/

ACPI_STATUS
AcpiUtCreateUpdateStateAndPush (
    ACPI_OPERAND_OBJECT     *Object,
    UINT16                  Action,
    ACPI_GENERIC_STATE      **StateList)
{
    ACPI_GENERIC_STATE      *State;


    ACPI_FUNCTION_ENTRY ();


    /* Ignore null objects; these are expected */

    if (!Object)
    {
        return (AE_OK);
    }

    State = AcpiUtCreateUpdateState (Object, Action);
    if (!State)
    {
        return (AE_NO_MEMORY);
    }

    AcpiUtPushGenericState (StateList, State);
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtWalkPackageTree
 *
 * PARAMETERS:  SourceObject        - The package to walk
 *              TargetObject        - Target object (if package is being copied)
 *              WalkCallback        - Called once for each package element
 *              Context             - Passed to the callback function
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk through a package
```

```
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtWalkPackageTree (
    ACPI_OPERAND_OBJECT     *SourceObject,
    void                    *TargetObject,
    ACPI_PKG_CALLBACK       WalkCallback,
    void                    *Context)
{
    ACPI_STATUS             Status = AE_OK;
    ACPI_GENERIC_STATE      *StateList = NULL;
    ACPI_GENERIC_STATE      *State;
    UINT32                  ThisIndex;
    ACPI_OPERAND_OBJECT     *ThisSourceObj;


    ACPI_FUNCTION_TRACE (UtWalkPackageTree);


    State = AcpiUtCreatePkgState (SourceObject, TargetObject, 0);
    if (!State)
    {
        return_ACPI_STATUS (AE_NO_MEMORY);
    }

    while (State)
    {
        /* Get one element of the package */

        ThisIndex     = State->Pkg.Index;
        ThisSourceObj = (ACPI_OPERAND_OBJECT *)
                        State->Pkg.SourceObject->Package.Elements[ThisIndex];

        /*
         * Check for:
         * 1) An uninitialized package element. It is completely
         *    legal to declare a package and leave it uninitialized
         * 2) Not an internal object - can be a namespace node instead
         * 3) Any type other than a package. Packages are handled in else
         *    case below.
         */
        if ((!ThisSourceObj) ||
            (ACPI_GET_DESCRIPTOR_TYPE (ThisSourceObj) != ACPI_DESC_TYPE_OPERAND) ||
            (ThisSourceObj->Common.Type != ACPI_TYPE_PACKAGE))
        {
            Status = WalkCallback (ACPI_COPY_TYPE_SIMPLE, ThisSourceObj,
                                   State, Context);
            if (ACPI_FAILURE (Status))
            {
                return_ACPI_STATUS (Status);
            }

            State->Pkg.Index++;
            while (State->Pkg.Index >= State->Pkg.SourceObject->Package.Count)
            {
                /*
                 * We've handled all of the objects at this level,  This means
                 * that we have just completed a package. That package may
                 * have contained one or more packages itself.
                 *
                 * Delete this state and pop the previous state (package).
                 */
                AcpiUtDeleteGenericState (State);
                State = AcpiUtPopGenericState (&StateList);
```



```
                /* Finished when there are no more states */

                if (!State)
                {
                    /*
                     * We have handled all of the objects in the top level
                     * package just add the length of the package objects
                     * and exit
                     */
                    return_ACPI_STATUS (AE_OK);
                }

                /*
                 * Go back up a level and move the index past the just
                 * completed package object.
                 */
                State->Pkg.Index++;
            }
        }
        else
        {
            /* This is a subobject of type package */

            Status = WalkCallback (ACPI_COPY_TYPE_PACKAGE, ThisSourceObj,
                                   State, Context);
            if (ACPI_FAILURE (Status))
            {
                return_ACPI_STATUS (Status);
            }

            /*
             * Push the current state and create a new one
             * The callback above returned a new target package object.
             */
            AcpiUtPushGenericState (&StateList, State);
            State = AcpiUtCreatePkgState (ThisSourceObj,
                                          State->Pkg.ThisTargetObj, 0);
            if (!State)
            {
                /* Free any stacked Update State objects */

                while (StateList)
                {
                    State = AcpiUtPopGenericState (&StateList);
                    AcpiUtDeleteGenericState (State);
                }
                return_ACPI_STATUS (AE_NO_MEMORY);
            }
        }
    }

    /* We should never get here */

    return_ACPI_STATUS (AE_AML_INTERNAL);
}


#ifdef ACPI_DEBUG_OUTPUT
/*****************************************************************************
 *
 * FUNCTION:    AcpiUtDisplayInitPathname
 *
 * PARAMETERS:  Type                - Object type of the node
 *              ObjHandle           - Handle whose pathname will be displayed
 *              Path                - Additional path string to be appended.
 *                                    (NULL if no extra path)
```

```
391  *
392  * RETURN:      ACPI_STATUS
393  *
394  * DESCRIPTION: Display full pathname of an object, DEBUG ONLY
395  *
396  ******************************************************************************/

398 void
399 AcpiUtDisplayInitPathname (
400     UINT8                   Type,
401     ACPI_NAMESPACE_NODE     *ObjHandle,
402     char                    *Path)
403 {
404     ACPI_STATUS             Status;
405     ACPI_BUFFER             Buffer;


408     ACPI_FUNCTION_ENTRY ();


411     /* Only print the path if the appropriate debug level is enabled */

413     if (!(AcpiDbgLevel & ACPI_LV_INIT_NAMES))
414     {
415         return;
416     }

418     /* Get the full pathname to the node */

420     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
421     Status = AcpiNsHandleToPathname (ObjHandle, &Buffer);
422     if (ACPI_FAILURE (Status))
423     {
424         return;
425     }

427     /* Print what we're doing */

429     switch (Type)
430     {
431     case ACPI_TYPE_METHOD:

433         AcpiOsPrintf ("Executing    ");
434         break;

436     default:

438         AcpiOsPrintf ("Initializing ");
439         break;
440     }

442     /* Print the object type and pathname */

444     AcpiOsPrintf ("%-12s  %s",
445         AcpiUtGetTypeName (Type), (char *) Buffer.Pointer);

447     /* Extra path is used to append names like _STA, _INI, etc. */

449     if (Path)
450     {
451         AcpiOsPrintf (".%s", Path);
452     }
453     AcpiOsPrintf ("\n");

455     ACPI_FREE (Buffer.Pointer);
456 }
```



```
457 #endif
```

****
11172 Thu Dec 26 13:49:41 2013
new/usr/src/common/acpica/components/utilities/utmutex.c
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Module Name: utmutex - local mutex support
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTMUTEX_C__

#include "acpi.h"
#include "accommon.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utmutex")

/* Local prototypes */

static ACPI_STATUS
AcpiUtCreateMutex (
    ACPI_MUTEX_HANDLE       MutexId);

static void
```



```
AcpiUtDeleteMutex (
    ACPI_MUTEX_HANDLE       MutexId);


/******************************************************************************
 *
 * FUNCTION:    AcpiUtMutexInitialize
 *
 * PARAMETERS:  None.
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create the system mutex objects. This includes mutexes,
 *              spin locks, and reader/writer locks.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtMutexInitialize (
    void)
{
    UINT32                  i;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (UtMutexInitialize);


    /* Create each of the predefined mutex objects */

    for (i = 0; i < ACPI_NUM_MUTEX; i++)
    {
        Status = AcpiUtCreateMutex (i);
        if (ACPI_FAILURE (Status))
        {
            return_ACPI_STATUS (Status);
        }
    }

    /* Create the spinlocks for use at interrupt level or for speed */
    /* Create the spinlocks for use at interrupt level */

    Status = AcpiOsCreateLock (&AcpiGbl_GpeLock);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    Status = AcpiOsCreateLock (&AcpiGbl_HardwareLock);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    Status = AcpiOsCreateLock (&AcpiGbl_ReferenceCountLock);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /* Mutex for _OSI support */

    Status = AcpiOsCreateMutex (&AcpiGbl_OsiMutex);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
```

```
125     }

127     /* Create the reader/writer lock for namespace access */

129     Status = AcpiUtCreateRwLock (&AcpiGbl_NamespaceRwLock);
130     return_ACPI_STATUS (Status);
131 }


134 /*******************************************************************************
135  *
136  * FUNCTION:    AcpiUtMutexTerminate
137  *
138  * PARAMETERS:  None.
139  *
140  * RETURN:      None.
141  *
142  * DESCRIPTION: Delete all of the system mutex objects. This includes mutexes,
143  *              spin locks, and reader/writer locks.
144  *
145  ******************************************************************************/

147 void
148 AcpiUtMutexTerminate (
149     void)
150 {
151     UINT32                  i;


154     ACPI_FUNCTION_TRACE (UtMutexTerminate);


157     /* Delete each predefined mutex object */

159     for (i = 0; i < ACPI_NUM_MUTEX; i++)
160     {
161         AcpiUtDeleteMutex (i);
162     }

164     AcpiOsDeleteMutex (AcpiGbl_OsiMutex);

166     /* Delete the spinlocks */

168     AcpiOsDeleteLock (AcpiGbl_GpeLock);
169     AcpiOsDeleteLock (AcpiGbl_HardwareLock);
170     AcpiOsDeleteLock (AcpiGbl_ReferenceCountLock);

172     /* Delete the reader/writer lock */

174     AcpiUtDeleteRwLock (&AcpiGbl_NamespaceRwLock);
175     return_VOID;
176 }
_____unchanged_portion_omitted_


212 /*******************************************************************************
213  *
214  * FUNCTION:    AcpiUtDeleteMutex
215  *
216  * PARAMETERS:  MutexID         - ID of the mutex to be deleted
217  *
218  * RETURN:      Status
219  *
220  * DESCRIPTION: Delete a mutex object.
221  *
222  ******************************************************************************/
```



```
224 static void
225 AcpiUtDeleteMutex (
226     ACPI_MUTEX_HANDLE       MutexId)
227 {

229     ACPI_FUNCTION_TRACE_U32 (UtDeleteMutex, MutexId);


232     AcpiOsDeleteMutex (AcpiGbl_MutexInfo[MutexId].Mutex);

234     AcpiGbl_MutexInfo[MutexId].Mutex = NULL;
235     AcpiGbl_MutexInfo[MutexId].ThreadId = ACPI_MUTEX_NOT_ACQUIRED;

237     return_VOID;
238 }
_____unchanged_portion_omitted_


332 /*******************************************************************************
333  *
334  * FUNCTION:    AcpiUtReleaseMutex
335  *
336  * PARAMETERS:  MutexID         - ID of the mutex to be released
337  *
338  * RETURN:      Status
339  *
340  * DESCRIPTION: Release a mutex object.
341  *
342  ******************************************************************************/

344 ACPI_STATUS
345 AcpiUtReleaseMutex (
346     ACPI_MUTEX_HANDLE       MutexId)
347 {
338     ACPI_THREAD_ID          ThisThreadId;


348     ACPI_FUNCTION_NAME (UtReleaseMutex);


344     ThisThreadId = AcpiOsGetThreadId ();
351     ACPI_DEBUG_PRINT ((ACPI_DB_MUTEX, "Thread %u releasing Mutex [%s]\n",
352         (UINT32) AcpiOsGetThreadId (), AcpiUtGetMutexName (MutexId)));
346         (UINT32) ThisThreadId, AcpiUtGetMutexName (MutexId)));

354     if (MutexId > ACPI_MAX_MUTEX)
355     {
356         return (AE_BAD_PARAMETER);
357     }

359     /*
360      * Mutex must be acquired in order to release it!
361      */
362     if (AcpiGbl_MutexInfo[MutexId].ThreadId == ACPI_MUTEX_NOT_ACQUIRED)
363     {
364         ACPI_ERROR ((AE_INFO,
365             "Mutex [0x%X] is not acquired, cannot release", MutexId));

367         return (AE_NOT_ACQUIRED);
368     }

370 #ifdef ACPI_MUTEX_DEBUG
371     {
372         UINT32                  i;
373         /*
```

```
374          * Mutex debug code, for internal debugging only.
375          *
376          * Deadlock prevention. Check if this thread owns any mutexes of value
377          * greater than this one. If so, the thread has violated the mutex
378          * ordering rule. This indicates a coding error somewhere in
379          * the ACPI subsystem code.
380          */
381         for (i = MutexId; i < ACPI_NUM_MUTEX; i++)
382         {
383             if (AcpiGbl_MutexInfo[i].ThreadId == AcpiOsGetThreadId ())
377             if (AcpiGbl_MutexInfo[i].ThreadId == ThisThreadId)
384             {
385                 if (i == MutexId)
386                 {
387                     continue;
388                 }
389
390                 ACPI_ERROR ((AE_INFO,
391                     "Invalid release order: owns [%s], releasing [%s]",
392                     AcpiUtGetMutexName (i), AcpiUtGetMutexName (MutexId)));
393
394                 return (AE_RELEASE_DEADLOCK);
395             }
396         }
397     }
398 #endif
399
400     /* Mark unlocked FIRST */
401
402     AcpiGbl_MutexInfo[MutexId].ThreadId = ACPI_MUTEX_NOT_ACQUIRED;
403
404     AcpiOsReleaseMutex (AcpiGbl_MutexInfo[MutexId].Mutex);
405     return (AE_OK);
406 }
```

```
**********************************************************
   21968 Thu Dec 26 13:49:42 2013
new/usr/src/common/acpica/components/utilities/utobject.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utobject - ACPI object create/delete/size/cache routines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #define __UTOBJECT_C__

  46 #include "acpi.h"
  47 #include "accommon.h"
  48 #include "acnamesp.h"


  51 #define _COMPONENT          ACPI_UTILITIES
  52         ACPI_MODULE_NAME    ("utobject")

  54 /* Local prototypes */

  56 static ACPI_STATUS
  57 AcpiUtGetSimpleObjectSize (
  58     ACPI_OPERAND_OBJECT     *Obj,
  59     ACPI_SIZE               *ObjLength);
```



```
  61 static ACPI_STATUS
  62 AcpiUtGetPackageObjectSize (
  63     ACPI_OPERAND_OBJECT     *Obj,
  64     ACPI_SIZE               *ObjLength);

  66 static ACPI_STATUS
  67 AcpiUtGetElementLength (
  68     UINT8                   ObjectType,
  69     ACPI_OPERAND_OBJECT     *SourceObject,
  70     ACPI_GENERIC_STATE      *State,
  71     void                    *Context);


  74 /******************************************************************************
  75  *
  76  * FUNCTION:    AcpiUtCreateInternalObjectDbg
  77  *
  78  * PARAMETERS:  ModuleName          - Source file name of caller
  79  *              LineNumber          - Line number of caller
  80  *              ComponentId         - Component type of caller
  81  *              Type                - ACPI Type of the new object
  82  *
  83  * RETURN:      A new internal object, null on failure
  84  *
  85  * DESCRIPTION: Create and initialize a new internal object.
  86  *
  87  * NOTE:        We always allocate the worst-case object descriptor because
  88  *              these objects are cached, and we want them to be
  89  *              one-size-satisifies-any-request. This in itself may not be
  90  *              the most memory efficient, but the efficiency of the object
  91  *              cache should more than make up for this!
  92  *
  93  *****************************************************************************/

  95 ACPI_OPERAND_OBJECT  *
  96 AcpiUtCreateInternalObjectDbg (
  97     const char              *ModuleName,
  98     UINT32                  LineNumber,
  99     UINT32                  ComponentId,
 100     ACPI_OBJECT_TYPE        Type)
 101 {
 102     ACPI_OPERAND_OBJECT     *Object;
 103     ACPI_OPERAND_OBJECT     *SecondObject;


 106     ACPI_FUNCTION_TRACE_STR (UtCreateInternalObjectDbg,
 107         AcpiUtGetTypeName (Type));


 110     /* Allocate the raw object descriptor */

 112     Object = AcpiUtAllocateObjectDescDbg (ModuleName, LineNumber, ComponentId);
 113     if (!Object)
 114     {
 115         return_PTR (NULL);
 116     }

 118     switch (Type)
 119     {
 120     case ACPI_TYPE_REGION:
 121     case ACPI_TYPE_BUFFER_FIELD:
 122     case ACPI_TYPE_LOCAL_BANK_FIELD:

 124         /* These types require a secondary object */
```

```
126         SecondObject = AcpiUtAllocateObjectDescDbg (ModuleName,
127                             LineNumber, ComponentId);
128         if (!SecondObject)
129         {
130             AcpiUtDeleteObjectDesc (Object);
131             return_PTR (NULL);
132         }
133
134         SecondObject->Common.Type = ACPI_TYPE_LOCAL_EXTRA;
135         SecondObject->Common.ReferenceCount = 1;
136
137         /* Link the second object to the first */
138
139         Object->Common.NextObject = SecondObject;
140         break;
141
142     default:
143
144         /* All others have no secondary object */
145         break;
146     }
147
148     /* Save the object type in the object descriptor */
149
150     Object->Common.Type = (UINT8) Type;
151
152     /* Init the reference count */
153
154     Object->Common.ReferenceCount = 1;
155
156     /* Any per-type initialization should go here */
157
158     return_PTR (Object);
159 }
```

unchanged_portion_omitted_

```
362 /*******************************************************************************
363  *
364  * FUNCTION:    AcpiUtValidInternalObject
365  *
366  * PARAMETERS:  Object              - Object to be validated
367  *
368  * RETURN:      TRUE if object is valid, FALSE otherwise
369  *
370  * DESCRIPTION: Validate a pointer to be of type ACPI_OPERAND_OBJECT
371  *
372  ******************************************************************************/
373
374 BOOLEAN
375 AcpiUtValidInternalObject (
376     void                    *Object)
377 {
378
379     ACPI_FUNCTION_NAME (UtValidInternalObject);
380
381
382     /* Check for a null pointer */
383
384     if (!Object)
385     {
386         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "**** Null Object Ptr\n"));
387         return (FALSE);
388     }
```



```
389
390     /* Check the descriptor type field */
391
392     switch (ACPI_GET_DESCRIPTOR_TYPE (Object))
393     {
394     case ACPI_DESC_TYPE_OPERAND:
395
396         /* The object appears to be a valid ACPI_OPERAND_OBJECT */
397
398         return (TRUE);
399
400     default:
401
402         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
403                 "%p is not not an ACPI operand obj [%s]\n",
404                 Object, AcpiUtGetDescriptorName (Object)));
405         break;
406     }
407
408     return (FALSE);
409 }
```

unchanged_portion_omitted_

```
459 /*******************************************************************************
460  *
461  * FUNCTION:    AcpiUtDeleteObjectDesc
462  *
463  * PARAMETERS:  Object          - An Acpi internal object to be deleted
464  *
465  * RETURN:      None.
466  *
467  * DESCRIPTION: Free an ACPI object descriptor or add it to the object cache
468  *
469  ******************************************************************************/
470
471 void
472 AcpiUtDeleteObjectDesc (
473     ACPI_OPERAND_OBJECT     *Object)
474 {
475     ACPI_FUNCTION_TRACE_PTR (UtDeleteObjectDesc, Object);
476
477     /* Object must be of type ACPI_OPERAND_OBJECT */
478     /* Object must be an ACPI_OPERAND_OBJECT  */
479
480     if (ACPI_GET_DESCRIPTOR_TYPE (Object) != ACPI_DESC_TYPE_OPERAND)
481     {
482         ACPI_ERROR ((AE_INFO,
483             "%p is not an ACPI Operand object [%s]", Object,
484             AcpiUtGetDescriptorName (Object)));
485         return_VOID;
486     }
487
488     (void) AcpiOsReleaseObject (AcpiGbl_OperandCache, Object);
489     return_VOID;
490 }
491
492
493 /*******************************************************************************
494  *
495  * FUNCTION:    AcpiUtGetSimpleObjectSize
496  *
497  * PARAMETERS:  InternalObject     - An ACPI operand object
498  *              ObjLength          - Where the length is returned
499  *
500  * RETURN:      Status
```

```
 501  *
 502  * DESCRIPTION: This function is called to determine the space required to
 503  *              contain a simple object for return to an external user.
 504  *
 505  *              The length includes the object structure plus any additional
 506  *              needed space.
 507  *
 508  ****************************************************************************/

 510 static ACPI_STATUS
 511 AcpiUtGetSimpleObjectSize (
 512     ACPI_OPERAND_OBJECT     *InternalObject,
 513     ACPI_SIZE               *ObjLength)
 514 {
 515     ACPI_SIZE               Length;
 516     ACPI_SIZE               Size;
 517     ACPI_STATUS             Status = AE_OK;


 520     ACPI_FUNCTION_TRACE_PTR (UtGetSimpleObjectSize, InternalObject);


 523     /* Start with the length of the (external) Acpi object */

 525     Length = sizeof (ACPI_OBJECT);

 527     /* A NULL object is allowed, can be a legal uninitialized package element */

 529     if (!InternalObject)
 530     {
 531         /*
 532          * Object is NULL, just return the length of ACPI_OBJECT
 533          * (A NULL ACPI_OBJECT is an object of all zeroes.)
 534          */
 535         *ObjLength = ACPI_ROUND_UP_TO_NATIVE_WORD (Length);
 536         return_ACPI_STATUS (AE_OK);
 537     }

 539     /* A Namespace Node should never appear here */

 541     if (ACPI_GET_DESCRIPTOR_TYPE (InternalObject) == ACPI_DESC_TYPE_NAMED)
 542     {
 543         /* A namespace node should never get here */

 545         return_ACPI_STATUS (AE_AML_INTERNAL);
 546     }

 548     /*
 549      * The final length depends on the object type
 550      * Strings and Buffers are packed right up against the parent object and
 551      * must be accessed bytewise or there may be alignment problems on
 552      * certain processors
 553      */
 554     switch (InternalObject->Common.Type)
 555     {
 556     case ACPI_TYPE_STRING:

 558         Length += (ACPI_SIZE) InternalObject->String.Length + 1;
 559         break;


 561     case ACPI_TYPE_BUFFER:

 563         Length += (ACPI_SIZE) InternalObject->Buffer.Length;
 564         break;
```



```
 566     case ACPI_TYPE_INTEGER:
 567     case ACPI_TYPE_PROCESSOR:
 568     case ACPI_TYPE_POWER:

 570         /* No extra data for these types */

 572         break;


 574     case ACPI_TYPE_LOCAL_REFERENCE:

 576         switch (InternalObject->Reference.Class)
 577         {
 578         case ACPI_REFCLASS_NAME:

 579             /*
 580              * Get the actual length of the full pathname to this object.
 581              * The reference will be converted to the pathname to the object
 582              */
 583             Size = AcpiNsGetPathnameLength (InternalObject->Reference.Node);
 584             if (!Size)
 585             {
 586                 return_ACPI_STATUS (AE_BAD_PARAMETER);
 587             }

 589             Length += ACPI_ROUND_UP_TO_NATIVE_WORD (Size);
 590             break;

 592         default:

 593             /*
 594              * No other reference opcodes are supported.
 595              * Notably, Locals and Args are not supported, but this may be
 596              * required eventually.
 597              */
 598             ACPI_ERROR ((AE_INFO, "Cannot convert to external object - "
 599                 "unsupported Reference Class [%s] 0x%X in object %p",
 600                 AcpiUtGetReferenceName (InternalObject),
 601                 InternalObject->Reference.Class, InternalObject));
 602             Status = AE_TYPE;
 603             break;
 604         }
 605         break;


 607     default:

 609         ACPI_ERROR ((AE_INFO, "Cannot convert to external object - "
 610             "unsupported type [%s] 0x%X in object %p",
 611             AcpiUtGetObjectTypeName (InternalObject),
 612             InternalObject->Common.Type, InternalObject));
 613         Status = AE_TYPE;
 614         break;
 615     }

 617     /*
 618      * Account for the space required by the object rounded up to the next
 619      * multiple of the machine word size. This keeps each object aligned
 620      * on a machine word boundary. (preventing alignment faults on some
 621      * machines.)
 622      */
 623     *ObjLength = ACPI_ROUND_UP_TO_NATIVE_WORD (Length);
 624     return_ACPI_STATUS (Status);
 625 }
```

```
 628 /*******************************************************************************
 629  *
 630  * FUNCTION:    AcpiUtGetElementLength
 631  *
 632  * PARAMETERS:  ACPI_PKG_CALLBACK
 633  *
 634  * RETURN:      Status
 635  *
 636  * DESCRIPTION: Get the length of one package element.
 637  *
 638  ******************************************************************************/

 640 static ACPI_STATUS
 641 AcpiUtGetElementLength (
 642     UINT8                   ObjectType,
 643     ACPI_OPERAND_OBJECT     *SourceObject,
 644     ACPI_GENERIC_STATE      *State,
 645     void                    *Context)
 646 {
 647     ACPI_STATUS             Status = AE_OK;
 648     ACPI_PKG_INFO           *Info = (ACPI_PKG_INFO *) Context;
 649     ACPI_SIZE               ObjectSpace;


 652     switch (ObjectType)
 653     {
 654     case ACPI_COPY_TYPE_SIMPLE:

 655         /*
 656          * Simple object - just get the size (Null object/entry is handled
 657          * here also) and sum it into the running package length
 658          */
 659         Status = AcpiUtGetSimpleObjectSize (SourceObject, &ObjectSpace);
 660         if (ACPI_FAILURE (Status))
 661         {
 662             return (Status);
 663         }

 665         Info->Length += ObjectSpace;
 666         break;


 668     case ACPI_COPY_TYPE_PACKAGE:

 670         /* Package object - nothing much to do here, let the walk handle it */

 672         Info->NumPackages++;
 673         State->Pkg.ThisTargetObj = NULL;
 674         break;


 676     default:

 678         /* No other types allowed */

 680         return (AE_BAD_PARAMETER);
 681     }

 683     return (Status);
 684 }
```

unchanged_portion_omitted_

```
**********************************************************
   15789 Thu Dec 26 13:49:42 2013
new/usr/src/common/acpica/components/utilities/utosi.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utosi - Support for the _OSI predefined control method
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTOSI_C__

#include "acpi.h"
#include "accommon.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utosi")

/*
 * Strings supported by the _OSI predefined control method (which is
 * implemented internally within this module.)
 *
 * March 2009: Removed "Linux" as this host no longer wants to respond true
 * for this string. Basically, the only safe OS strings are windows-related
 * and in many or most cases represent the only test path within the
```



```
 * BIOS-provided ASL code.
 *
 * The last element of each entry is used to track the newest version of
 * Windows that the BIOS has requested.
 */
static ACPI_INTERFACE_INFO    AcpiDefaultSupportedInterfaces[] =
{
    /* Operating System Vendor Strings */

    {"Windows 2000",        NULL, 0, ACPI_OSI_WIN_2000},         /* Windows 2000
    {"Windows 2001",        NULL, 0, ACPI_OSI_WIN_XP},           /* Windows XP *
    {"Windows 2001 SP1",    NULL, 0, ACPI_OSI_WIN_XP_SP1},       /* Windows XP S
    {"Windows 2001.1",      NULL, 0, ACPI_OSI_WINSRV_2003},      /* Windows Serv
    {"Windows 2001 SP2",    NULL, 0, ACPI_OSI_WIN_XP_SP2},       /* Windows XP S
    {"Windows 2001.1 SP1",  NULL, 0, ACPI_OSI_WINSRV_2003_SP1},  /* Windows Serv
    {"Windows 2006",        NULL, 0, ACPI_OSI_WIN_VISTA},        /* Windows Vist
    {"Windows 2006.1",      NULL, 0, ACPI_OSI_WINSRV_2008},      /* Windows Serv
    {"Windows 2006 SP1",    NULL, 0, ACPI_OSI_WIN_VISTA_SP1},    /* Windows Vist
    {"Windows 2006 SP2",    NULL, 0, ACPI_OSI_WIN_VISTA_SP2},    /* Windows Vist
    {"Windows 2009",        NULL, 0, ACPI_OSI_WIN_7},            /* Windows 7 an
    {"Windows 2012",        NULL, 0, ACPI_OSI_WIN_8},            /* Windows 8 an

    /* Feature Group Strings */

    {"Extended Address Space Descriptor", NULL, ACPI_OSI_FEATURE, 0},
    {"Extended Address Space Descriptor", NULL, 0, 0}

    /*
     * All "optional" feature group strings (features that are implemented
     * by the host) should be dynamically modified to VALID by the host via
     * AcpiInstallInterface or AcpiUpdateInterfaces. Such optional feature
     * group strings are set as INVALID by default here.
     * by the host) should be dynamically added by the host via
     * AcpiInstallInterface and should not be manually added here.
     *
     * Examples of optional feature group strings:
     *
     * "Module Device"
     * "Processor Device"
     * "3.0 Thermal Model"
     * "3.0 _SCP Extensions"
     * "Processor Aggregator Device"
     */

    {"Module Device",               NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
    {"Processor Device",            NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
    {"3.0 Thermal Model",           NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
    {"3.0 _SCP Extensions",         NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
    {"Processor Aggregator Device", NULL, ACPI_OSI_OPTIONAL_FEATURE, 0}
};


/******************************************************************************
 *
 * FUNCTION:    AcpiUtInitializeInterfaces
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize the global _OSI supported interfaces list
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtInitializeInterfaces (
```

```
    void)
{
    ACPI_STATUS             Status;
    UINT32                  i;


    Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
    AcpiGbl_SupportedInterfaces = AcpiDefaultSupportedInterfaces;

    /* Link the static list of supported interfaces */

    for (i = 0; i < (ACPI_ARRAY_LENGTH (AcpiDefaultSupportedInterfaces) - 1); i+
    {
        AcpiDefaultSupportedInterfaces[i].Next =
            &AcpiDefaultSupportedInterfaces[(ACPI_SIZE) i + 1];
    }

    AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtInterfaceTerminate
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 * RETURN:      None
 *
 * DESCRIPTION: Delete all interfaces in the global list. Sets
 *              AcpiGbl_SupportedInterfaces to NULL.
 *
 *****************************************************************************/

ACPI_STATUS
void
AcpiUtInterfaceTerminate (
    void)
{
    ACPI_STATUS             Status;
    ACPI_INTERFACE_INFO     *NextInterface;


    Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
    NextInterface = AcpiGbl_SupportedInterfaces;

    while (NextInterface)
    {
        AcpiGbl_SupportedInterfaces = NextInterface->Next;

        if (NextInterface->Flags & ACPI_OSI_DYNAMIC)
        {
```



```
            /* Only interfaces added at runtime can be freed */

        if (NextInterface->Flags & ACPI_OSI_DYNAMIC)
        {
            ACPI_FREE (NextInterface->Name);
            ACPI_FREE (NextInterface);
        }
        else
        {
            /* Interface is in static list. Reset it to invalid or valid. */

            if (NextInterface->Flags & ACPI_OSI_DEFAULT_INVALID)
            {
                NextInterface->Flags |= ACPI_OSI_INVALID;
            }
            else
            {
                NextInterface->Flags &= ~ACPI_OSI_INVALID;
            }
        }

        NextInterface = AcpiGbl_SupportedInterfaces;
    }

    AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
    return (AE_OK);
}
```

*unchanged_portion_omitted_*

```
/******************************************************************************
 *
 * FUNCTION:    AcpiUtUpdateInterfaces
 *
 * PARAMETERS:  Action              - Actions to be performed during the
 *                                    update
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Update _OSI interface strings, disabling or enabling OS vendor
 *              strings or/and feature group strings.
 *              Caller MUST hold AcpiGbl_OsiMutex
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtUpdateInterfaces (
    UINT8                   Action)
{
    ACPI_INTERFACE_INFO     *NextInterface;


    NextInterface = AcpiGbl_SupportedInterfaces;
    while (NextInterface)
    {
        if (((NextInterface->Flags & ACPI_OSI_FEATURE) &&
             (Action & ACPI_FEATURE_STRINGS)) ||
            (!(NextInterface->Flags & ACPI_OSI_FEATURE) &&
             (Action & ACPI_VENDOR_STRINGS)))
        {
            if (Action & ACPI_DISABLE_INTERFACES)
            {
                /* Mark the interfaces as invalid */

                NextInterface->Flags |= ACPI_OSI_INVALID;
            }
```

```
356             else
357             {
358                 /* Mark the interfaces as valid */

360                 NextInterface->Flags &= ~ACPI_OSI_INVALID;
361             }
362         }

364         NextInterface = NextInterface->Next;
365     }

367     return (AE_OK);
368 }


371 /*******************************************************************************
372  *
373  * FUNCTION:    AcpiUtGetInterface
374  *
375  * PARAMETERS:  InterfaceName       - The interface to find
376  *
377  * RETURN:      ACPI_INTERFACE_INFO if found. NULL if not found.
378  *
379  * DESCRIPTION: Search for the specified interface name in the global list.
380  *              Caller MUST hold AcpiGbl_OsiMutex
381  *
382  ******************************************************************************/

384 ACPI_INTERFACE_INFO *
385 AcpiUtGetInterface (
386     ACPI_STRING             InterfaceName)
387 {
388     ACPI_INTERFACE_INFO     *NextInterface;


391     NextInterface = AcpiGbl_SupportedInterfaces;
392     while (NextInterface)
393     {
394         if (!ACPI_STRCMP (InterfaceName, NextInterface->Name))
395         {
396             return (NextInterface);
397         }

399         NextInterface = NextInterface->Next;
400     }

402     return (NULL);
403 }


406 /*******************************************************************************
407  *
408  * FUNCTION:    AcpiUtOsiImplementation
409  *
410  * PARAMETERS:  WalkState           - Current walk state
411  *
412  * RETURN:      Status
413  *
414  * DESCRIPTION: Implementation of the _OSI predefined control method. When
415  *              an invocation of _OSI is encountered in the system AML,
416  *              control is transferred to this function.
417  *
418  ******************************************************************************/

420 ACPI_STATUS
421 AcpiUtOsiImplementation (
```



```
422     ACPI_WALK_STATE         *WalkState)
423 {
424     ACPI_OPERAND_OBJECT     *StringDesc;
425     ACPI_OPERAND_OBJECT     *ReturnDesc;
426     ACPI_INTERFACE_INFO     *InterfaceInfo;
427     ACPI_INTERFACE_HANDLER  InterfaceHandler;
428     ACPI_STATUS             Status;
429     UINT32                  ReturnValue;


432     ACPI_FUNCTION_TRACE (UtOsiImplementation);


435     /* Validate the string input argument (from the AML caller) */

437     StringDesc = WalkState->Arguments[0].Object;
438     if (!StringDesc ||
439         (StringDesc->Common.Type != ACPI_TYPE_STRING))
440     {
441         return_ACPI_STATUS (AE_TYPE);
442     }

444     /* Create a return object */

446     ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
447     if (!ReturnDesc)
448     {
449         return_ACPI_STATUS (AE_NO_MEMORY);
450     }

452     /* Default return value is 0, NOT SUPPORTED */

454     ReturnValue = 0;
455     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
456     if (ACPI_FAILURE (Status))
457     {
458         AcpiUtRemoveReference (ReturnDesc);
459         return_ACPI_STATUS (Status);
460     }
378     (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);

462     /* Lookup the interface in the global _OSI list */

464     InterfaceInfo = AcpiUtGetInterface (StringDesc->String.Pointer);
465     if (InterfaceInfo &&
466         !(InterfaceInfo->Flags & ACPI_OSI_INVALID))
467     {
468         /*
469          * The interface is supported.
470          * Update the OsiData if necessary. We keep track of the latest
471          * version of Windows that has been requested by the BIOS.
472          */
473         if (InterfaceInfo->Value > AcpiGbl_OsiData)
474         {
475             AcpiGbl_OsiData = InterfaceInfo->Value;
476         }

478         ReturnValue = ACPI_UINT32_MAX;
479     }

481     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);

483     /*
484      * Invoke an optional _OSI interface handler. The host OS may wish
485      * to do some interface-specific handling. For example, warn about
486      * certain interfaces or override the true/false support value.
```

```
 487      */
 488     InterfaceHandler = AcpiGbl_InterfaceHandler;
 489     if (InterfaceHandler)
 490     {
 491         ReturnValue = InterfaceHandler (
 492             StringDesc->String.Pointer, ReturnValue);
 493     }

 495     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INFO,
 496         "ACPI: BIOS _OSI(\"%s\") is %ssupported\n",
 497         StringDesc->String.Pointer, ReturnValue == 0 ? "not " : ""));

 499     /* Complete the return object */

 501     ReturnDesc->Integer.Value = ReturnValue;
 502     WalkState->ReturnDesc = ReturnDesc;
 503     return_ACPI_STATUS (AE_OK);
 504 }
```

unchanged_portion_omitted_

```
**********************************************************
    7652 Thu Dec 26 13:49:42 2013
new/usr/src/common/acpica/components/utilities/utownerid.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utownerid - Support for Table/Method Owner IDs
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTOWNERID_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utownerid")


/******************************************************************************
 *
 * FUNCTION:    AcpiUtAllocateOwnerId
 *
 * PARAMETERS:  OwnerId         - Where the new owner ID is returned
 *
```



```
 * RETURN:      Status
 *
 * DESCRIPTION: Allocate a table or method owner ID. The owner ID is used to
 *              track objects created by the table or method, to be deleted
 *              when the method exits or the table is unloaded.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiUtAllocateOwnerId (
    ACPI_OWNER_ID           *OwnerId)
{
    UINT32                  i;
    UINT32                  j;
    UINT32                  k;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (UtAllocateOwnerId);


    /* Guard against multiple allocations of ID to the same location */

    if (*OwnerId)
    {
        ACPI_ERROR ((AE_INFO, "Owner ID [0x%2.2X] already exists", *OwnerId));
        return_ACPI_STATUS (AE_ALREADY_EXISTS);
    }

    /* Mutex for the global ID mask */

    Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * Find a free owner ID, cycle through all possible IDs on repeated
     * allocations. (ACPI_NUM_OWNERID_MASKS + 1) because first index may have
     * to be scanned twice.
     */
    for (i = 0, j = AcpiGbl_LastOwnerIdIndex;
         i < (ACPI_NUM_OWNERID_MASKS + 1);
         i++, j++)
    {
        if (j >= ACPI_NUM_OWNERID_MASKS)
        {
            j = 0;  /* Wraparound to start of mask array */
        }

        for (k = AcpiGbl_NextOwnerIdOffset; k < 32; k++)
        {
            if (AcpiGbl_OwnerIdMask[j] == ACPI_UINT32_MAX)
            {
                /* There are no free IDs in this mask */

                break;
            }

            if (!(AcpiGbl_OwnerIdMask[j] & (1 << k)))
            {
                /*
                 * Found a free ID. The actual ID is the bit index plus one,
                 * making zero an invalid Owner ID. Save this as the last ID
                 * allocated and update the global ID mask.
```

```
128                  */
129                 AcpiGbl_OwnerIdMask[j] |= (1 << k);
130
131                 AcpiGbl_LastOwnerIdIndex = (UINT8) j;
132                 AcpiGbl_NextOwnerIdOffset = (UINT8) (k + 1);
133
134                 /*
135                  * Construct encoded ID from the index and bit position
136                  *
137                  * Note: Last [j].k (bit 255) is never used and is marked
138                  * permanently allocated (prevents +1 overflow)
139                  */
140                 *OwnerId = (ACPI_OWNER_ID) ((k + 1) + ACPI_MUL_32 (j));
141
142                 ACPI_DEBUG_PRINT ((ACPI_DB_VALUES,
143                     "Allocated OwnerId: %2.2X\n", (unsigned int) *OwnerId));
144                 goto Exit;
145             }
146         }
147
148         AcpiGbl_NextOwnerIdOffset = 0;
149     }
150
151     /*
152      * All OwnerIds have been allocated. This typically should
153      * not happen since the IDs are reused after deallocation. The IDs are
154      * allocated upon table load (one per table) and method execution, and
155      * they are released when a table is unloaded or a method completes
156      * execution.
157      *
158      * If this error happens, there may be very deep nesting of invoked control
159      * methods, or there may be a bug where the IDs are not released.
160      */
161     Status = AE_OWNER_ID_LIMIT;
162     ACPI_ERROR ((AE_INFO,
163         "Could not allocate new OwnerId (255 max), AE_OWNER_ID_LIMIT"));
164
165 Exit:
166     (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
167     return_ACPI_STATUS (Status);
168 }
169
170
171 /*******************************************************************************
172  *
173  * FUNCTION:    AcpiUtReleaseOwnerId
174  *
175  * PARAMETERS:  OwnerIdPtr          - Pointer to a previously allocated OwnerID
176  *
177  * RETURN:      None. No error is returned because we are either exiting a
178  *              control method or unloading a table. Either way, we would
179  *              ignore any error anyway.
180  *
181  * DESCRIPTION: Release a table or method owner ID. Valid IDs are 1 - 255
182  *
183  ******************************************************************************/
184
185 void
186 AcpiUtReleaseOwnerId (
187     ACPI_OWNER_ID           *OwnerIdPtr)
188 {
189     ACPI_OWNER_ID           OwnerId = *OwnerIdPtr;
190     ACPI_STATUS             Status;
191     UINT32                  Index;
192     UINT32                  Bit;
```



```
193
194
195     ACPI_FUNCTION_TRACE_U32 (UtReleaseOwnerId, OwnerId);
196
197
198     /* Always clear the input OwnerId (zero is an invalid ID) */
199
200     *OwnerIdPtr = 0;
201
202     /* Zero is not a valid OwnerID */
203
204     if (OwnerId == 0)
205     {
206         ACPI_ERROR ((AE_INFO, "Invalid OwnerId: 0x%2.2X", OwnerId));
207         return_VOID;
208     }
209
210     /* Mutex for the global ID mask */
211
212     Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
213     if (ACPI_FAILURE (Status))
214     {
215         return_VOID;
216     }
217
218     /* Normalize the ID to zero */
219
220     OwnerId--;
221
222     /* Decode ID to index/offset pair */
223
224     Index = ACPI_DIV_32 (OwnerId);
225     Bit = 1 << ACPI_MOD_32 (OwnerId);
226
227     /* Free the owner ID only if it is valid */
228
229     if (AcpiGbl_OwnerIdMask[Index] & Bit)
230     {
231         AcpiGbl_OwnerIdMask[Index] ^= Bit;
232     }
233     else
234     {
235         ACPI_ERROR ((AE_INFO,
236             "Release of non-allocated OwnerId: 0x%2.2X", OwnerId + 1));
237     }
238
239     (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
240     return_VOID;
241 }
```

```
**********************************************************
   12498 Thu Dec 26 13:49:43 2013
new/usr/src/common/acpica/components/utilities/utpredef.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utpredef - support functions for predefined names
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTPREDEF_C__

#include "acpi.h"
#include "accommon.h"
#include "acpredef.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utpredef")


/*
 * Names for the types that can be returned by the predefined objects.
 * Used for warning messages. Must be in the same order as the ACPI_RTYPEs
 */
static const char   *UtRtypeNames[] =
{
    "/Integer",
    "/String",
    "/Buffer",
    "/Package",
    "/Reference",
};


/******************************************************************************
 *
 * FUNCTION:    AcpiUtGetNextPredefinedMethod
 *
 * PARAMETERS:  ThisName            - Entry in the predefined method/name table
 *
 * RETURN:      Pointer to next entry in predefined table.
 *
 * DESCRIPTION: Get the next entry in the predefine method table. Handles the
 *              cases where a package info entry follows a method name that
 *              returns a package.
 *
 *****************************************************************************/

const ACPI_PREDEFINED_INFO *
AcpiUtGetNextPredefinedMethod (
    const ACPI_PREDEFINED_INFO  *ThisName)
{

    /*
     * Skip next entry in the table if this name returns a Package
     * (next entry contains the package info)
     */
    if ((ThisName->Info.ExpectedBtypes & ACPI_RTYPE_PACKAGE) &&
        (ThisName->Info.ExpectedBtypes != ACPI_RTYPE_ALL))
    {
        ThisName++;
    }

    ThisName++;
    return (ThisName);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtMatchPredefinedMethod
 *
 * PARAMETERS:  Name                - Name to find
 *
 * RETURN:      Pointer to entry in predefined table. NULL indicates not found.
 *
 * DESCRIPTION: Check an object name against the predefined object list.
 *
 *****************************************************************************/

const ACPI_PREDEFINED_INFO *
AcpiUtMatchPredefinedMethod (
    char                        *Name)
{
    const ACPI_PREDEFINED_INFO  *ThisName;


    /* Quick check for a predefined name, first character must be underscore */

    if (Name[0] != '_')
    {
        return (NULL);
    }
```

```
    /* Search info table for a predefined method/object name */

    ThisName = AcpiGbl_PredefinedMethods;
    while (ThisName->Info.Name[0])
    {
        if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
        {
            return (ThisName);
        }

        ThisName = AcpiUtGetNextPredefinedMethod (ThisName);
    }

    return (NULL); /* Not found */
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtGetExpectedReturnTypes
 *
 * PARAMETERS:  Buffer              - Where the formatted string is returned
 *              ExpectedBTypes      - Bitfield of expected data types
 *
 * RETURN:      Formatted string in Buffer.
 *
 * DESCRIPTION: Format the expected object types into a printable string.
 *
 ******************************************************************************/

void
AcpiUtGetExpectedReturnTypes (
    char                    *Buffer,
    UINT32                  ExpectedBtypes)
{
    UINT32                  ThisRtype;
    UINT32                  i;
    UINT32                  j;


    if (!ExpectedBtypes)
    {
        ACPI_STRCPY (Buffer, "NONE");
        return;
    }

    j = 1;
    Buffer[0] = 0;
    ThisRtype = ACPI_RTYPE_INTEGER;

    for (i = 0; i < ACPI_NUM_RTYPES; i++)
    {
        /* If one of the expected types, concatenate the name of this type */

        if (ExpectedBtypes & ThisRtype)
        {
            ACPI_STRCAT (Buffer, &UtRtypeNames[i][j]);
            j = 0;              /* Use name separator from now on */
        }

        ThisRtype <<= 1;    /* Next Rtype */
    }
}
```



```
/*******************************************************************************
 *
 * The remaining functions are used by iASL and AcpiHelp only
 *
 ******************************************************************************/

#if (defined ACPI_ASL_COMPILER || defined ACPI_HELP_APP)
#include <stdio.h>
#include <string.h>

/* Local prototypes */

static UINT32
AcpiUtGetArgumentTypes (
    char                    *Buffer,
    UINT16                  ArgumentTypes);


/* Types that can be returned externally by a predefined name */

static const char   *UtExternalTypeNames[] = /* Indexed by ACPI_TYPE_* */
{
    ", UNSUPPORTED-TYPE",
    ", Integer",
    ", String",
    ", Buffer",
    ", Package"
};

/* Bit widths for resource descriptor predefined names */

static const char   *UtResourceTypeNames[] =
{
    "/1",
    "/2",
    "/3",
    "/8",
    "/16",
    "/32",
    "/64",
    "/variable",
};


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtMatchResourceName
 *
 * PARAMETERS:  Name                - Name to find
 *
 * RETURN:      Pointer to entry in the resource table. NULL indicates not
 *              found.
 *
 * DESCRIPTION: Check an object name against the predefined resource
 *              descriptor object list.
 *
 ******************************************************************************/

const ACPI_PREDEFINED_INFO *
AcpiUtMatchResourceName (
    char                        *Name)
{
    const ACPI_PREDEFINED_INFO  *ThisName;


    /* Quick check for a predefined name, first character must be underscore */
```

```
261     if (Name[0] != '_')
262     {
263         return (NULL);
264     }

266     /* Search info table for a predefined method/object name */

268     ThisName = AcpiGbl_ResourceNames;
269     while (ThisName->Info.Name[0])
270     {
271         if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
272         {
273             return (ThisName);
274         }

276         ThisName++;
277     }

279     return (NULL); /* Not found */
280 }


283 /******************************************************************************
284  *
285  * FUNCTION:    AcpiUtDisplayPredefinedMethod
286  *
287  * PARAMETERS:  Buffer              - Scratch buffer for this function
288  *              ThisName            - Entry in the predefined method/name table
289  *              MultiLine           - TRUE if output should be on >1 line
290  *
291  * RETURN:      None
292  *
293  * DESCRIPTION: Display information about a predefined method. Number and
294  *              type of the input arguments, and expected type(s) for the
295  *              return value, if any.
296  *
297  *****************************************************************************/

299 void
300 AcpiUtDisplayPredefinedMethod (
301     char                    *Buffer,
302     const ACPI_PREDEFINED_INFO  *ThisName,
303     BOOLEAN                 MultiLine)
304 {
305     UINT32                  ArgCount;

307     /*
308      * Get the argument count and the string buffer
309      * containing all argument types
310      */
311     ArgCount = AcpiUtGetArgumentTypes (Buffer,
312         ThisName->Info.ArgumentList);

314     if (MultiLine)
315     {
316         printf ("      ");
317     }

319     printf ("%4.4s    Requires %s%u argument%s",
320         ThisName->Info.Name,
321         (ThisName->Info.ArgumentList & ARG_COUNT_IS_MINIMUM) ?
322             "(at least) " : "",
323         ArgCount, ArgCount != 1 ? "s" : "");

325     /* Display the types for any arguments */
```



```
327     if (ArgCount > 0)
328     {
329         printf (" (%s)", Buffer);
330     }

332     if (MultiLine)
333     {
334         printf ("\n    ");
335     }

337     /* Get the return value type(s) allowed */

339     if (ThisName->Info.ExpectedBtypes)
340     {
341         AcpiUtGetExpectedReturnTypes (Buffer, ThisName->Info.ExpectedBtypes);
342         printf ("  Return value types: %s\n", Buffer);
343     }
344     else
345     {
346         printf ("  No return value\n");
347     }
348 }


351 /******************************************************************************
352  *
353  * FUNCTION:    AcpiUtGetArgumentTypes
354  *
355  * PARAMETERS:  Buffer              - Where to return the formatted types
356  *              ArgumentTypes       - Types field for this method
357  *
358  * RETURN:      Count - the number of arguments required for this method
359  *
360  * DESCRIPTION: Format the required data types for this method (Integer,
361  *              String, Buffer, or Package) and return the required argument
362  *              count.
363  *
364  *****************************************************************************/

366 static UINT32
367 AcpiUtGetArgumentTypes (
368     char                    *Buffer,
369     UINT16                  ArgumentTypes)
370 {
371     UINT16                  ThisArgumentType;
372     UINT16                  SubIndex;
373     UINT16                  ArgCount;
374     UINT32                  i;


377     *Buffer = 0;
378     SubIndex = 2;

380     /* First field in the types list is the count of args to follow */

382     ArgCount = METHOD_GET_ARG_COUNT (ArgumentTypes);
383     if (ArgCount > METHOD_PREDEF_ARGS_MAX)
384     {
385         printf ("**** Invalid argument count (%u) "
386             "in predefined info structure\n", ArgCount);
387         return (ArgCount);
388     }

390     /* Get each argument from the list, convert to ascii, store to buffer */
```

```
392     for (i = 0; i < ArgCount; i++)
393     {
394         ThisArgumentType = METHOD_GET_NEXT_TYPE (ArgumentTypes);

396         if (!ThisArgumentType || (ThisArgumentType > METHOD_MAX_ARG_TYPE))
397         {
398             printf ("**** Invalid argument type (%u) "
399                 "in predefined info structure\n", ThisArgumentType);
400             return (ArgCount);
401         }

403         strcat (Buffer, UtExternalTypeNames[ThisArgumentType] + SubIndex);
404         SubIndex = 0;
405     }

407     return (ArgCount);
408 }


411 /*******************************************************************************
412  *
413  * FUNCTION:    AcpiUtGetResourceBitWidth
414  *
415  * PARAMETERS:  Buffer              - Where the formatted string is returned
416  *              Types               - Bitfield of expected data types
417  *
418  * RETURN:      Count of return types. Formatted string in Buffer.
419  *
420  * DESCRIPTION: Format the resource bit widths into a printable string.
421  *
422  ******************************************************************************/

424 UINT32
425 AcpiUtGetResourceBitWidth (
426     char                    *Buffer,
427     UINT16                  Types)
428 {
429     UINT32                  i;
430     UINT16                  SubIndex;
431     UINT32                  Found;


434     *Buffer = 0;
435     SubIndex = 1;
436     Found = 0;

438     for (i = 0; i < NUM_RESOURCE_WIDTHS; i++)
439     {
440         if (Types & 1)
441         {
442             strcat (Buffer, &(UtResourceTypeNames[i][SubIndex]));
443             SubIndex = 0;
444             Found++;
445         }

447         Types >>= 1;
448     }

450     return (Found);
451 }
452 #endif
```

```
**********************************************************
   24427 Thu Dec 26 13:49:43 2013
new/usr/src/common/acpica/components/utilities/utresrc.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utresrc - Resource management utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTRESRC_C__

#include "acpi.h"
#include "accommon.h"
#include "acresrc.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utresrc")


#if defined(ACPI_DISASSEMBLER) || defined (ACPI_DEBUGGER)

/*
 * Strings used to decode resource descriptors.
 * Used by both the disassembler and the debugger resource dump routines
```



```
 */
const char                      *AcpiGbl_BmDecode[] =
{
    "NotBusMaster",
    "BusMaster"
};

const char                      *AcpiGbl_ConfigDecode[] =
{
    "0 - Good Configuration",
    "1 - Acceptable Configuration",
    "2 - Suboptimal Configuration",
    "3 - ***Invalid Configuration***",
};

const char                      *AcpiGbl_ConsumeDecode[] =
{
    "ResourceProducer",
    "ResourceConsumer"
};

const char                      *AcpiGbl_DecDecode[] =
{
    "PosDecode",
    "SubDecode"
};

const char                      *AcpiGbl_HeDecode[] =
{
    "Level",
    "Edge"
};

const char                      *AcpiGbl_IoDecode[] =
{
    "Decode10",
    "Decode16"
};

const char                      *AcpiGbl_LlDecode[] =
{
    "ActiveHigh",
    "ActiveLow"
};

const char                      *AcpiGbl_MaxDecode[] =
{
    "MaxNotFixed",
    "MaxFixed"
};

const char                      *AcpiGbl_MemDecode[] =
{
    "NonCacheable",
    "Cacheable",
    "WriteCombining",
    "Prefetchable"
};

const char                      *AcpiGbl_MinDecode[] =
{
    "MinNotFixed",
    "MinFixed"
};

const char                      *AcpiGbl_MtpDecode[] =
```

```
127 {
128     "AddressRangeMemory",
129     "AddressRangeReserved",
130     "AddressRangeACPI",
131     "AddressRangeNVS"
132 };

134 const char                      *AcpiGbl_RngDecode[] =
135 {
136     "InvalidRanges",
137     "NonISAOnlyRanges",
138     "ISAOnlyRanges",
139     "EntireRange"
140 };

142 const char                      *AcpiGbl_RwDecode[] =
143 {
144     "ReadOnly",
145     "ReadWrite"
146 };

148 const char                      *AcpiGbl_ShrDecode[] =
149 {
150     "Exclusive",
151     "Shared",
152     "ExclusiveAndWake",         /* ACPI 5.0 */
153     "SharedAndWake"             /* ACPI 5.0 */
154 };

156 const char                      *AcpiGbl_SizDecode[] =
157 {
158     "Transfer8",
159     "Transfer8_16",
160     "Transfer16",
161     "InvalidSize"
162 };

164 const char                      *AcpiGbl_TrsDecode[] =
165 {
166     "DenseTranslation",
167     "SparseTranslation"
168 };

170 const char                      *AcpiGbl_TtpDecode[] =
171 {
172     "TypeStatic",
173     "TypeTranslation"
174 };

176 const char                      *AcpiGbl_TypDecode[] =
177 {
178     "Compatibility",
179     "TypeA",
180     "TypeB",
181     "TypeF"
182 };

184 const char                      *AcpiGbl_PpcDecode[] =
185 {
186     "PullDefault",
187     "PullUp",
188     "PullDown",
189     "PullNone"
190 };

192 const char                      *AcpiGbl_IorDecode[] =
```



```
193 {
194     "IoRestrictionNone",
195     "IoRestrictionInputOnly",
196     "IoRestrictionOutputOnly",
197     "IoRestrictionNoneAndPreserve"
198 };

200 const char                      *AcpiGbl_DtsDecode[] =
201 {
202     "Width8bit",
203     "Width16bit",
204     "Width32bit",
205     "Width64bit",
206     "Width128bit",
207     "Width256bit",
208 };

210 /* GPIO connection type */

212 const char                      *AcpiGbl_CtDecode[] =
213 {
214     "Interrupt",
215     "I/O"
216 };

218 /* Serial bus type */

220 const char                      *AcpiGbl_SbtDecode[] =
221 {
222     "/* UNKNOWN serial bus type */",
223     "I2C",
224     "SPI",
225     "UART"
226 };

228 /* I2C serial bus access mode */

230 const char                      *AcpiGbl_AmDecode[] =
231 {
232     "AddressingMode7Bit",
233     "AddressingMode10Bit"
234 };

236 /* I2C serial bus slave mode */

238 const char                      *AcpiGbl_SmDecode[] =
239 {
240     "ControllerInitiated",
241     "DeviceInitiated"
242 };

244 /* SPI serial bus wire mode */

246 const char                      *AcpiGbl_WmDecode[] =
247 {
248     "FourWireMode",
249     "ThreeWireMode"
250 };

252 /* SPI serial clock phase */

254 const char                      *AcpiGbl_CphDecode[] =
255 {
256     "ClockPhaseFirst",
257     "ClockPhaseSecond"
258 };
```

```
260 /* SPI serial bus clock polarity */

262 const char                     *AcpiGbl_CpoDecode[] =
263 {
264     "ClockPolarityLow",
265     "ClockPolarityHigh"
266 };

268 /* SPI serial bus device polarity */

270 const char                     *AcpiGbl_DpDecode[] =
271 {
272     "PolarityLow",
273     "PolarityHigh"
274 };

276 /* UART serial bus endian */

278 const char                     *AcpiGbl_EdDecode[] =
279 {
280     "LittleEndian",
281     "BigEndian"
282 };

284 /* UART serial bus bits per byte */

286 const char                     *AcpiGbl_BpbDecode[] =
287 {
288     "DataBitsFive",
289     "DataBitsSix",
290     "DataBitsSeven",
291     "DataBitsEight",
292     "DataBitsNine",
293     "/* UNKNOWN Bits per byte */",
294     "/* UNKNOWN Bits per byte */",
295     "/* UNKNOWN Bits per byte */"
296 };

298 /* UART serial bus stop bits */

300 const char                     *AcpiGbl_SbDecode[] =
301 {
302     "StopBitsNone",
303     "StopBitsOne",
304     "StopBitsOnePlusHalf",
305     "StopBitsTwo"
306 };

308 /* UART serial bus flow control */

310 const char                     *AcpiGbl_FcDecode[] =
311 {
312     "FlowControlNone",
313     "FlowControlHardware",
314     "FlowControlXON",
315     "/* UNKNOWN flow control keyword */"
316 };

318 /* UART serial bus parity type */

320 const char                     *AcpiGbl_PtDecode[] =
321 {
322     "ParityTypeNone",
323     "ParityTypeEven",
324     "ParityTypeOdd",
```



```
325     "ParityTypeMark",
326     "ParityTypeSpace",
327     "/* UNKNOWN parity keyword */",
328     "/* UNKNOWN parity keyword */",
329     "/* UNKNOWN parity keyword */"
330 };

332 #endif


335 /*
336  * Base sizes of the raw AML resource descriptors, indexed by resource type.
337  * Zero indicates a reserved (and therefore invalid) resource type.
338  */
339 const UINT8                 AcpiGbl_ResourceAmlSizes[] =
340 {
341     /* Small descriptors */

343     0,
344     0,
345     0,
346     0,
347     ACPI_AML_SIZE_SMALL (AML_RESOURCE_IRQ),
348     ACPI_AML_SIZE_SMALL (AML_RESOURCE_DMA),
349     ACPI_AML_SIZE_SMALL (AML_RESOURCE_START_DEPENDENT),
350     ACPI_AML_SIZE_SMALL (AML_RESOURCE_END_DEPENDENT),
351     ACPI_AML_SIZE_SMALL (AML_RESOURCE_IO),
352     ACPI_AML_SIZE_SMALL (AML_RESOURCE_FIXED_IO),
353     ACPI_AML_SIZE_SMALL (AML_RESOURCE_FIXED_DMA),
354     0,
355     0,
356     0,
357     ACPI_AML_SIZE_SMALL (AML_RESOURCE_VENDOR_SMALL),
358     ACPI_AML_SIZE_SMALL (AML_RESOURCE_END_TAG),

360     /* Large descriptors */

362     0,
363     ACPI_AML_SIZE_LARGE (AML_RESOURCE_MEMORY24),
364     ACPI_AML_SIZE_LARGE (AML_RESOURCE_GENERIC_REGISTER),
365     0,
366     ACPI_AML_SIZE_LARGE (AML_RESOURCE_VENDOR_LARGE),
367     ACPI_AML_SIZE_LARGE (AML_RESOURCE_MEMORY32),
368     ACPI_AML_SIZE_LARGE (AML_RESOURCE_FIXED_MEMORY32),
369     ACPI_AML_SIZE_LARGE (AML_RESOURCE_ADDRESS32),
370     ACPI_AML_SIZE_LARGE (AML_RESOURCE_ADDRESS16),
371     ACPI_AML_SIZE_LARGE (AML_RESOURCE_EXTENDED_IRQ),
372     ACPI_AML_SIZE_LARGE (AML_RESOURCE_ADDRESS64),
373     ACPI_AML_SIZE_LARGE (AML_RESOURCE_EXTENDED_ADDRESS64),
374     ACPI_AML_SIZE_LARGE (AML_RESOURCE_GPIO),
375     0,
376     ACPI_AML_SIZE_LARGE (AML_RESOURCE_COMMON_SERIALBUS),
377 };

379 const UINT8                 AcpiGbl_ResourceAmlSerialBusSizes[] =
380 {
381     0,
382     ACPI_AML_SIZE_LARGE (AML_RESOURCE_I2C_SERIALBUS),
383     ACPI_AML_SIZE_LARGE (AML_RESOURCE_SPI_SERIALBUS),
384     ACPI_AML_SIZE_LARGE (AML_RESOURCE_UART_SERIALBUS),
385 };


388 /*
389  * Resource types, used to validate the resource length field.
390  * The length of fixed-length types must match exactly, variable
```

```
 * lengths must meet the minimum required length, etc.
 * Zero indicates a reserved (and therefore invalid) resource type.
 */
static const UINT8          AcpiGbl_ResourceTypes[] =
{
    /* Small descriptors */

    0,
    0,
    0,
    0,
    ACPI_SMALL_VARIABLE_LENGTH,     /* 04 IRQ */
    ACPI_FIXED_LENGTH,              /* 05 DMA */
    ACPI_SMALL_VARIABLE_LENGTH,     /* 06 StartDependentFunctions */
    ACPI_FIXED_LENGTH,              /* 07 EndDependentFunctions */
    ACPI_FIXED_LENGTH,              /* 08 IO */
    ACPI_FIXED_LENGTH,              /* 09 FixedIO */
    ACPI_FIXED_LENGTH,              /* 0A FixedDMA */
    0,
    0,
    0,
    ACPI_VARIABLE_LENGTH,           /* 0E VendorShort */
    ACPI_FIXED_LENGTH,              /* 0F EndTag */

    /* Large descriptors */

    0,
    ACPI_FIXED_LENGTH,              /* 01 Memory24 */
    ACPI_FIXED_LENGTH,              /* 02 GenericRegister */
    0,
    ACPI_VARIABLE_LENGTH,           /* 04 VendorLong */
    ACPI_FIXED_LENGTH,              /* 05 Memory32 */
    ACPI_FIXED_LENGTH,              /* 06 Memory32Fixed */
    ACPI_VARIABLE_LENGTH,           /* 07 Dword* address */
    ACPI_VARIABLE_LENGTH,           /* 08 Word* address */
    ACPI_VARIABLE_LENGTH,           /* 09 ExtendedIRQ */
    ACPI_VARIABLE_LENGTH,           /* 0A Qword* address */
    ACPI_FIXED_LENGTH,              /* 0B Extended* address */
    ACPI_VARIABLE_LENGTH,           /* 0C Gpio* */
    0,
    ACPI_VARIABLE_LENGTH            /* 0E *SerialBus */
};


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtWalkAmlResources
 *
 * PARAMETERS:  WalkState           - Current walk info
 * PARAMETERS:  Aml                 - Pointer to the raw AML resource template
 *              AmlLength           - Length of the entire template
 *              UserFunction        - Called once for each descriptor found. If
 *                                    NULL, a pointer to the EndTag is returned
 *              Context             - Passed to UserFunction
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Walk a raw AML resource list(buffer). User function called
 *              once for each resource found.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiUtWalkAmlResources (
    ACPI_WALK_STATE         *WalkState,
    UINT8                   *Aml,
```



```
    ACPI_SIZE               AmlLength,
    ACPI_WALK_AML_CALLBACK  UserFunction,
    void                    **Context)
{
    ACPI_STATUS             Status;
    UINT8                   *EndAml;
    UINT8                   ResourceIndex;
    UINT32                  Length;
    UINT32                  Offset = 0;
    UINT8                   EndTag[2] = {0x79, 0x00};


    ACPI_FUNCTION_TRACE (UtWalkAmlResources);


    /* The absolute minimum resource template is one EndTag descriptor */

    if (AmlLength < sizeof (AML_RESOURCE_END_TAG))
    {
        return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
    }

    /* Point to the end of the resource template buffer */

    EndAml = Aml + AmlLength;

    /* Walk the byte list, abort on any invalid descriptor type or length */

    while (Aml < EndAml)
    {
        /* Validate the Resource Type and Resource Length */

        Status = AcpiUtValidateResource (WalkState, Aml, &ResourceIndex);
        if (ACPI_FAILURE (Status))
        {
            /*
             * Exit on failure. Cannot continue because the descriptor length
             * may be bogus also.
             */
            return_ACPI_STATUS (Status);
        }

        /* Get the length of this descriptor */

        Length = AcpiUtGetDescriptorLength (Aml);

        /* Invoke the user function */

        if (UserFunction)
        {
            Status = UserFunction (Aml, Length, Offset, ResourceIndex, Context);
            if (ACPI_FAILURE (Status))
            {
                return_ACPI_STATUS (Status);
            }
        }

        /* An EndTag descriptor terminates this resource template */

        if (AcpiUtGetResourceType (Aml) == ACPI_RESOURCE_NAME_END_TAG)
        {
            /*
             * There must be at least one more byte in the buffer for
             * the 2nd byte of the EndTag
             */
            if ((Aml + 1) >= EndAml)
```

```
523             {
524                 return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
525             }

527             /* Return the pointer to the EndTag if requested */

529             if (!UserFunction)
530             {
531                 *Context = Aml;
532             }

534             /* Normal exit */

536             return_ACPI_STATUS (AE_OK);
537         }

539         Aml += Length;
540         Offset += Length;
541     }

543     /* Did not find an EndTag descriptor */

545     if (UserFunction)
546     {
547         /* Insert an EndTag anyway. AcpiRsGetListLength always leaves room */

549         (void) AcpiUtValidateResource (WalkState, EndTag, &ResourceIndex);
550         Status = UserFunction (EndTag, 2, Offset, ResourceIndex, Context);
551         if (ACPI_FAILURE (Status))
552         {
553             return_ACPI_STATUS (Status);
554         }
555     }

557     return_ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
558 }


561 /*******************************************************************************
562  *
563  * FUNCTION:    AcpiUtValidateResource
564  *
565  * PARAMETERS:  WalkState           - Current walk info
566  *              Aml                 - Pointer to the raw AML resource descriptor
567  *              ReturnIndex         - Where the resource index is returned. NULL
568  *                                    if the index is not required.
569  *
570  * RETURN:      Status, and optionally the Index into the global resource tables
571  *
572  * DESCRIPTION: Validate an AML resource descriptor by checking the Resource
573  *              Type and Resource Length. Returns an index into the global
574  *              resource information/dispatch tables for later use.
575  *
576  ******************************************************************************/

578 ACPI_STATUS
579 AcpiUtValidateResource (
580     ACPI_WALK_STATE         *WalkState,
581     void                    *Aml,
582     UINT8                   *ReturnIndex)
583 {
584     AML_RESOURCE            *AmlResource;
585     UINT8                   ResourceType;
586     UINT8                   ResourceIndex;
587     ACPI_RS_LENGTH          ResourceLength;
588     ACPI_RS_LENGTH          MinimumResourceLength;
```



```
591     ACPI_FUNCTION_ENTRY ();


594     /*
595      * 1) Validate the ResourceType field (Byte 0)
596      */
597     ResourceType = ACPI_GET8 (Aml);

599     /*
600      * Byte 0 contains the descriptor name (Resource Type)
601      * Examine the large/small bit in the resource header
602      */
603     if (ResourceType & ACPI_RESOURCE_NAME_LARGE)
604     {
605         /* Verify the large resource type (name) against the max */

607         if (ResourceType > ACPI_RESOURCE_NAME_LARGE_MAX)
608         {
609             goto InvalidResource;
610         }

612         /*
613          * Large Resource Type -- bits 6:0 contain the name
614          * Translate range 0x80-0x8B to index range 0x10-0x1B
615          */
616         ResourceIndex = (UINT8) (ResourceType - 0x70);
617     }
618     else
619     {
620         /*
621          * Small Resource Type -- bits 6:3 contain the name
622          * Shift range to index range 0x00-0x0F
623          */
624         ResourceIndex = (UINT8)
625             ((ResourceType & ACPI_RESOURCE_NAME_SMALL_MASK) >> 3);
626     }

628     /*
629      * Check validity of the resource type, via AcpiGbl_ResourceTypes. Zero
630      * indicates an invalid resource.
631      */
632     if (!AcpiGbl_ResourceTypes[ResourceIndex])
633     {
634         goto InvalidResource;
635     }

637     /*
638      * Validate the ResourceLength field. This ensures that the length
639      * is at least reasonable, and guarantees that it is non-zero.
640      */
641     ResourceLength = AcpiUtGetResourceLength (Aml);
642     MinimumResourceLength = AcpiGbl_ResourceAmlSizes[ResourceIndex];

644     /* Validate based upon the type of resource - fixed length or variable */

646     switch (AcpiGbl_ResourceTypes[ResourceIndex])
647     {
648     case ACPI_FIXED_LENGTH:

650         /* Fixed length resource, length must match exactly */

652         if (ResourceLength != MinimumResourceLength)
653         {
654             goto BadResourceLength;
```

```
655         }
656         break;

658     case ACPI_VARIABLE_LENGTH:

660         /* Variable length resource, length must be at least the minimum */

662         if (ResourceLength < MinimumResourceLength)
663         {
664             goto BadResourceLength;
665         }
666         break;

668     case ACPI_SMALL_VARIABLE_LENGTH:

670         /* Small variable length resource, length can be (Min) or (Min-1) */

672         if ((ResourceLength > MinimumResourceLength) ||
673             (ResourceLength < (MinimumResourceLength - 1)))
674         {
675             goto BadResourceLength;
676         }
677         break;

679     default:

681         /* Shouldn't happen (because of validation earlier), but be sure */

683         goto InvalidResource;
684     }

686     AmlResource = ACPI_CAST_PTR (AML_RESOURCE, Aml);
687     if (ResourceType == ACPI_RESOURCE_NAME_SERIAL_BUS)
688     {
689         /* Validate the BusType field */

691         if ((AmlResource->CommonSerialBus.Type == 0) ||
692             (AmlResource->CommonSerialBus.Type > AML_RESOURCE_MAX_SERIALBUSTYPE))
693         {
694             if (WalkState)
695             {
696                 ACPI_ERROR ((AE_INFO,
697                     "Invalid/unsupported SerialBus resource descriptor: BusType
698                     AmlResource->CommonSerialBus.Type));
699             }
700             return (AE_AML_INVALID_RESOURCE_TYPE);
701         }
702     }

704     /* Optionally return the resource table index */

706     if (ReturnIndex)
707     {
708         *ReturnIndex = ResourceIndex;
709     }

711     return (AE_OK);


714 InvalidResource:

716     if (WalkState)
717     {
718         ACPI_ERROR ((AE_INFO,
719             "Invalid/unsupported resource descriptor: Type 0x%2.2X",
720             ResourceType));
```



```
721     }
722     return (AE_AML_INVALID_RESOURCE_TYPE);

724 BadResourceLength:

726     if (WalkState)
727     {
728         ACPI_ERROR ((AE_INFO,
729             "Invalid resource descriptor length: Type "
730             "0x%2.2X, Length 0x%4.4X, MinLength 0x%4.4X",
731             ResourceType, ResourceLength, MinimumResourceLength));
732     }
733     return (AE_AML_BAD_RESOURCE_LENGTH);
734 }


737 /*******************************************************************************
738  *
739  * FUNCTION:    AcpiUtGetResourceType
740  *
741  * PARAMETERS:  Aml             - Pointer to the raw AML resource descriptor
742  *
743  * RETURN:      The Resource Type with no extraneous bits (except the
744  *              Large/Small descriptor bit -- this is left alone)
745  *
746  * DESCRIPTION: Extract the Resource Type/Name from the first byte of
747  *              a resource descriptor.
748  *
749  ******************************************************************************/

751 UINT8
752 AcpiUtGetResourceType (
753     void                    *Aml)
754 {
755     ACPI_FUNCTION_ENTRY ();


758     /*
759      * Byte 0 contains the descriptor name (Resource Type)
760      * Examine the large/small bit in the resource header
761      */
762     if (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_LARGE)
763     {
764         /* Large Resource Type -- bits 6:0 contain the name */

766         return (ACPI_GET8 (Aml));
767     }
768     else
769     {
770         /* Small Resource Type -- bits 6:3 contain the name */

772         return ((UINT8) (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_SMALL_MASK));
773     }
774 }


777 /*******************************************************************************
778  *
779  * FUNCTION:    AcpiUtGetResourceLength
780  *
781  * PARAMETERS:  Aml             - Pointer to the raw AML resource descriptor
782  *
783  * RETURN:      Byte Length
784  *
785  * DESCRIPTION: Get the "Resource Length" of a raw AML descriptor. By
786  *              definition, this does not include the size of the descriptor
```

```
 *              header or the length field itself.
 *
 ******************************************************************************/

UINT16
AcpiUtGetResourceLength (
    void                    *Aml)
{
    ACPI_RS_LENGTH          ResourceLength;


    ACPI_FUNCTION_ENTRY ();


    /*
     * Byte 0 contains the descriptor name (Resource Type)
     * Examine the large/small bit in the resource header
     */
    if (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_LARGE)
    {
        /* Large Resource type -- bytes 1-2 contain the 16-bit length */

        ACPI_MOVE_16_TO_16 (&ResourceLength, ACPI_ADD_PTR (UINT8, Aml, 1));

    }
    else
    {
        /* Small Resource type -- bits 2:0 of byte 0 contain the length */

        ResourceLength = (UINT16) (ACPI_GET8 (Aml) &
                                   ACPI_RESOURCE_NAME_SMALL_LENGTH_MASK);
    }

    return (ResourceLength);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtGetResourceHeaderLength
 *
 * PARAMETERS:  Aml             - Pointer to the raw AML resource descriptor
 *
 * RETURN:      Length of the AML header (depends on large/small descriptor)
 *
 * DESCRIPTION: Get the length of the header for this resource.
 *
 ******************************************************************************/

UINT8
AcpiUtGetResourceHeaderLength (
    void                    *Aml)
{
    ACPI_FUNCTION_ENTRY ();


    /* Examine the large/small bit in the resource header */

    if (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_LARGE)
    {
        return (sizeof (AML_RESOURCE_LARGE_HEADER));
    }
    else
    {
        return (sizeof (AML_RESOURCE_SMALL_HEADER));
    }
```



```
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtGetDescriptorLength
 *
 * PARAMETERS:  Aml             - Pointer to the raw AML resource descriptor
 *
 * RETURN:      Byte length
 *
 * DESCRIPTION: Get the total byte length of a raw AML descriptor, including the
 *              length of the descriptor header and the length field itself.
 *              Used to walk descriptor lists.
 *
 ******************************************************************************/

UINT32
AcpiUtGetDescriptorLength (
    void                    *Aml)
{
    ACPI_FUNCTION_ENTRY ();


    /*
     * Get the Resource Length (does not include header length) and add
     * the header length (depends on if this is a small or large resource)
     */
    return (AcpiUtGetResourceLength (Aml) +
            AcpiUtGetResourceHeaderLength (Aml));
}


/******************************************************************************
 *
 * FUNCTION:    AcpiUtGetResourceEndTag
 *
 * PARAMETERS:  ObjDesc         - The resource template buffer object
 *              EndTag          - Where the pointer to the EndTag is returned
 *
 * RETURN:      Status, pointer to the end tag
 *
 * DESCRIPTION: Find the EndTag resource descriptor in an AML resource template
 *              Note: allows a buffer length of zero.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiUtGetResourceEndTag (
    ACPI_OPERAND_OBJECT     *ObjDesc,
    UINT8                   **EndTag)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (UtGetResourceEndTag);


    /* Allow a buffer length of zero */

    if (!ObjDesc->Buffer.Length)
    {
        *EndTag = ObjDesc->Buffer.Pointer;
        return_ACPI_STATUS (AE_OK);
    }
```

```
    /* Validate the template and get a pointer to the EndTag */

    Status = AcpiUtWalkAmlResources (NULL, ObjDesc->Buffer.Pointer,
                ObjDesc->Buffer.Length, NULL, (void **) EndTag);

    return_ACPI_STATUS (Status);
}
```

****
10412 Thu Dec 26 13:49:43 2013
new/usr/src/common/acpica/components/utilities/utstate.c
acpica-unix2-20130823
PANKOVs restructure
****

```
   1 /******************************************************************************
   2  *
   3  * Module Name: utstate - state object support procedures
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */


  45 #define __UTSTATE_C__

  47 #include "acpi.h"
  48 #include "accommon.h"

  50 #define _COMPONENT          ACPI_UTILITIES
  51         ACPI_MODULE_NAME    ("utstate")


  54 /*******************************************************************************
  55  *
  56  * FUNCTION:    AcpiUtCreatePkgStateAndPush
  57  *
  58  * PARAMETERS:  Object          - Object to be added to the new state
  59  *              Action          - Increment/Decrement
```



```
  60  *              StateList       - List the state will be added to
  61  *
  62  * RETURN:      Status
  63  *
  64  * DESCRIPTION: Create a new state and push it
  65  *
  66  ******************************************************************************/

  68 ACPI_STATUS
  69 AcpiUtCreatePkgStateAndPush (
  70     void                    *InternalObject,
  71     void                    *ExternalObject,
  72     UINT16                  Index,
  73     ACPI_GENERIC_STATE      **StateList)
  74 {
  75     ACPI_GENERIC_STATE      *State;


  78     ACPI_FUNCTION_ENTRY ();


  81     State = AcpiUtCreatePkgState (InternalObject, ExternalObject, Index);
  82     if (!State)
  83     {
  84         return (AE_NO_MEMORY);
  85     }

  87     AcpiUtPushGenericState (StateList, State);
  88     return (AE_OK);
  89 }


  92 /*******************************************************************************
  93  *
  94  * FUNCTION:    AcpiUtPushGenericState
  95  *
  96  * PARAMETERS:  ListHead            - Head of the state stack
  97  *              State               - State object to push
  98  *
  99  * RETURN:      None
 100  *
 101  * DESCRIPTION: Push a state object onto a state stack
 102  *
 103  ******************************************************************************/

 105 void
 106 AcpiUtPushGenericState (
 107     ACPI_GENERIC_STATE      **ListHead,
 108     ACPI_GENERIC_STATE      *State)
 109 {
 110     ACPI_FUNCTION_ENTRY ();
 110     ACPI_FUNCTION_TRACE (UtPushGenericState);


 113     /* Push the state object onto the front of the list (stack) */

 115     State->Common.Next = *ListHead;
 116     *ListHead = State;
 117     return;

 118     return_VOID;
 118 }


 121 /*******************************************************************************
 122  *
```

```
 123  * FUNCTION:    AcpiUtPopGenericState
 124  *
 125  * PARAMETERS:  ListHead            - Head of the state stack
 126  *
 127  * RETURN:      The popped state object
 128  *
 129  * DESCRIPTION: Pop a state object from a state stack
 130  *
 131  ******************************************************************************/

 133 ACPI_GENERIC_STATE *
 134 AcpiUtPopGenericState (
 135     ACPI_GENERIC_STATE      **ListHead)
 136 {
 137     ACPI_GENERIC_STATE      *State;


 140     ACPI_FUNCTION_ENTRY ();
 141     ACPI_FUNCTION_TRACE (UtPopGenericState);


 143     /* Remove the state object at the head of the list (stack) */

 145     State = *ListHead;
 146     if (State)
 147     {
 148         /* Update the list head */

 150         *ListHead = State->Common.Next;
 151     }

 153     return (State);
 154     return_PTR (State);
 154 }
_____unchanged_portion_omitted_


 191 /******************************************************************************
 192  *
 193  * FUNCTION:    AcpiUtCreateThreadState
 194  *
 195  * PARAMETERS:  None
 196  *
 197  * RETURN:      New Thread State. NULL on failure
 198  *
 199  * DESCRIPTION: Create a "Thread State" - a flavor of the generic state used
 200  *              to track per-thread info during method execution
 201  *
 202  ******************************************************************************/

 204 ACPI_THREAD_STATE *
 205 AcpiUtCreateThreadState (
 206     void)
 207 {
 208     ACPI_GENERIC_STATE      *State;


 211     ACPI_FUNCTION_ENTRY ();
 212     ACPI_FUNCTION_TRACE (UtCreateThreadState);


 214     /* Create the generic state object */

 216     State = AcpiUtCreateGenericState ();
 217     if (!State)
 218     {
```



```
 219         return (NULL);
 220         return_PTR (NULL);
 220     }

 222     /* Init fields specific to the update struct */

 224     State->Common.DescriptorType = ACPI_DESC_TYPE_STATE_THREAD;
 225     State->Thread.ThreadId = AcpiOsGetThreadId ();

 227     /* Check for invalid thread ID - zero is very bad, it will break things */

 229     if (!State->Thread.ThreadId)
 230     {
 231         ACPI_ERROR ((AE_INFO, "Invalid zero ID from AcpiOsGetThreadId"));
 232         State->Thread.ThreadId = (ACPI_THREAD_ID) 1;
 233     }

 235     return ((ACPI_THREAD_STATE *) State);
 236     return_PTR ((ACPI_THREAD_STATE *) State);
 236 }


 239 /******************************************************************************
 240  *
 241  * FUNCTION:    AcpiUtCreateUpdateState
 242  *
 243  * PARAMETERS:  Object          - Initial Object to be installed in the state
 244  *              Action          - Update action to be performed
 245  *
 246  * RETURN:      New state object, null on failure
 247  *
 248  * DESCRIPTION: Create an "Update State" - a flavor of the generic state used
 249  *              to update reference counts and delete complex objects such
 250  *              as packages.
 251  *
 252  ******************************************************************************/

 254 ACPI_GENERIC_STATE *
 255 AcpiUtCreateUpdateState (
 256     ACPI_OPERAND_OBJECT     *Object,
 257     UINT16                  Action)
 258 {
 259     ACPI_GENERIC_STATE      *State;


 262     ACPI_FUNCTION_ENTRY ();
 263     ACPI_FUNCTION_TRACE_PTR (UtCreateUpdateState, Object);


 265     /* Create the generic state object */

 267     State = AcpiUtCreateGenericState ();
 268     if (!State)
 269     {
 270         return (NULL);
 271         return_PTR (NULL);
 271     }

 273     /* Init fields specific to the update struct */

 275     State->Common.DescriptorType = ACPI_DESC_TYPE_STATE_UPDATE;
 276     State->Update.Object = Object;
 277     State->Update.Value = Action;
 278     return (State);

 280     return_PTR (State);
```

```
 279 }


 282 /*******************************************************************************
 283  *
 284  * FUNCTION:    AcpiUtCreatePkgState
 285  *
 286  * PARAMETERS:  Object          - Initial Object to be installed in the state
 287  *              Action          - Update action to be performed
 288  *
 289  * RETURN:      New state object, null on failure
 290  *
 291  * DESCRIPTION: Create a "Package State"
 292  *
 293  ******************************************************************************/

 295 ACPI_GENERIC_STATE *
 296 AcpiUtCreatePkgState (
 297     void                    *InternalObject,
 298     void                    *ExternalObject,
 299     UINT16                  Index)
 300 {
 301     ACPI_GENERIC_STATE      *State;


 304     ACPI_FUNCTION_ENTRY ();
 306     ACPI_FUNCTION_TRACE_PTR (UtCreatePkgState, InternalObject);


 307     /* Create the generic state object */

 309     State = AcpiUtCreateGenericState ();
 310     if (!State)
 311     {
 312         return (NULL);
 314         return_PTR (NULL);
 313     }

 315     /* Init fields specific to the update struct */

 317     State->Common.DescriptorType = ACPI_DESC_TYPE_STATE_PACKAGE;
 318     State->Pkg.SourceObject = (ACPI_OPERAND_OBJECT *) InternalObject;
 319     State->Pkg.DestObject = ExternalObject;
 320     State->Pkg.Index= Index;
 321     State->Pkg.NumPackages = 1;
 322     return (State);

 325     return_PTR (State);
 323 }


 326 /*******************************************************************************
 327  *
 328  * FUNCTION:    AcpiUtCreateControlState
 329  *
 330  * PARAMETERS:  None
 331  *
 332  * RETURN:      New state object, null on failure
 333  *
 334  * DESCRIPTION: Create a "Control State" - a flavor of the generic state used
 335  *              to support nested IF/WHILE constructs in the AML.
 336  *
 337  ******************************************************************************/

 339 ACPI_GENERIC_STATE *
 340 AcpiUtCreateControlState (
```



```
 341     void)
 342 {
 343     ACPI_GENERIC_STATE      *State;


 346     ACPI_FUNCTION_ENTRY ();
 349     ACPI_FUNCTION_TRACE (UtCreateControlState);


 349     /* Create the generic state object */

 351     State = AcpiUtCreateGenericState ();
 352     if (!State)
 353     {
 354         return (NULL);
 357         return_PTR (NULL);
 355     }

 357     /* Init fields specific to the control struct */

 359     State->Common.DescriptorType = ACPI_DESC_TYPE_STATE_CONTROL;
 360     State->Common.State = ACPI_CONTROL_CONDITIONAL_EXECUTING;
 361     return (State);

 365     return_PTR (State);
 362 }


 365 /*******************************************************************************
 366  *
 367  * FUNCTION:    AcpiUtDeleteGenericState
 368  *
 369  * PARAMETERS:  State               - The state object to be deleted
 370  *
 371  * RETURN:      None
 372  *
 373  * DESCRIPTION: Release a state object to the state cache. NULL state objects
 374  *              are ignored.
 375  *
 376  ******************************************************************************/

 378 void
 379 AcpiUtDeleteGenericState (
 380     ACPI_GENERIC_STATE      *State)
 381 {
 382     ACPI_FUNCTION_ENTRY ();
 386     ACPI_FUNCTION_TRACE (UtDeleteGenericState);


 385     /* Ignore null state */

 387     if (State)
 388     {
 389         (void) AcpiOsReleaseObject (AcpiGbl_StateCache, State);
 390     }
 391     return;
 395     return_VOID;
 392 }
```

```
**********************************************************
   19423 Thu Dec 26 13:49:44 2013
new/usr/src/common/acpica/components/utilities/utstring.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utstring - Common functions for strings and characters
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTSTRING_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utstring")


/*
 * Non-ANSI C library functions - strlwr, strupr, stricmp, and a 64-bit
 * version of strtoul.
 */
```



```
#ifdef ACPI_ASL_COMPILER
/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrlwr (strlwr)
 *
 * PARAMETERS:  SrcString       - The source string to convert
 *
 * RETURN:      None
 *
 * DESCRIPTION: Convert string to lowercase
 *
 * NOTE: This is not a POSIX function, so it appears here, not in utclib.c
 *
 ******************************************************************************/

void
AcpiUtStrlwr (
    char                    *SrcString)
{
    char                    *String;


    ACPI_FUNCTION_ENTRY ();


    if (!SrcString)
    {
        return;
    }

    /* Walk entire string, lowercasing the letters */

    for (String = SrcString; *String; String++)
    {
        *String = (char) ACPI_TOLOWER (*String);
    }

    return;
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStricmp (stricmp)
 *
 * PARAMETERS:  String1             - first string to compare
 *              String2             - second string to compare
 *
 * RETURN:      int that signifies string relationship. Zero means strings
 *              are equal.
 *
 * DESCRIPTION: Implementation of the non-ANSI stricmp function (compare
 *              strings with no case sensitivity)
 *
 ******************************************************************************/

int
AcpiUtStricmp (
    char                    *String1,
    char                    *String2)
{
    int                     c1;
    int                     c2;


    do
```

```
    {
        c1 = tolower ((int) *String1);
        c2 = tolower ((int) *String2);

        String1++;
        String2++;
    }
    while ((c1 == c2) && (c1));

    return (c1 - c2);
}
#endif


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrupr (strupr)
 *
 * PARAMETERS:  SrcString       - The source string to convert
 *
 * RETURN:      None
 *
 * DESCRIPTION: Convert string to uppercase
 *
 * NOTE: This is not a POSIX function, so it appears here, not in utclib.c
 *
 ******************************************************************************/

void
AcpiUtStrupr (
    char                    *SrcString)
{
    char                    *String;


    ACPI_FUNCTION_ENTRY ();


    if (!SrcString)
    {
        return;
    }

    /* Walk entire string, uppercasing the letters */

    for (String = SrcString; *String; String++)
    {
        *String = (char) ACPI_TOUPPER (*String);
    }

    return;
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtStrtoul64
 *
 * PARAMETERS:  String          - Null terminated string
 *              Base            - Radix of the string: 16 or ACPI_ANY_BASE;
 *                                ACPI_ANY_BASE means 'in behalf of ToInteger'
 *              RetInteger      - Where the converted integer is returned
 *
 * RETURN:      Status and Converted value
 *
 * DESCRIPTION: Convert a string into an unsigned value. Performs either a
```



```
 *              32-bit or 64-bit conversion, depending on the current mode
 *              of the interpreter.
 *              NOTE: Does not support Octal strings, not needed.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiUtStrtoul64 (
    char                    *String,
    UINT32                  Base,
    UINT64                  *RetInteger)
{
    UINT32                  ThisDigit = 0;
    UINT64                  ReturnValue = 0;
    UINT64                  Quotient;
    UINT64                  Dividend;
    UINT32                  ToIntegerOp = (Base == ACPI_ANY_BASE);
    UINT32                  Mode32 = (AcpiGbl_IntegerByteWidth == 4);
    UINT8                   ValidDigits = 0;
    UINT8                   SignOf0x = 0;
    UINT8                   Term = 0;


    ACPI_FUNCTION_TRACE_STR (UtStroul64, String);


    switch (Base)
    {
    case ACPI_ANY_BASE:
    case 16:

        break;

    default:

        /* Invalid Base */

        return_ACPI_STATUS (AE_BAD_PARAMETER);
    }

    if (!String)
    {
        goto ErrorExit;
    }

    /* Skip over any white space in the buffer */

    while ((*String) && (ACPI_IS_SPACE (*String) || *String == '\t'))
    {
        String++;
    }

    if (ToIntegerOp)
    {
        /*
         * Base equal to ACPI_ANY_BASE means 'ToInteger operation case'.
         * We need to determine if it is decimal or hexadecimal.
         */
        if ((*String == '0') && (ACPI_TOLOWER (*(String + 1)) == 'x'))
        {
            SignOf0x = 1;
            Base = 16;

            /* Skip over the leading '0x' */
            String += 2;
        }
```

```
259         else
260         {
261             Base = 10;
262         }
263     }

265     /* Any string left? Check that '0x' is not followed by white space. */

267     if (!(*String) || ACPI_IS_SPACE (*String) || *String == '\t')
268     {
269         if (ToIntegerOp)
270         {
271             goto ErrorExit;
272         }
273         else
274         {
275             goto AllDone;
276         }
277     }

279     /*
280      * Perform a 32-bit or 64-bit conversion, depending upon the current
281      * execution mode of the interpreter
282      */
283     Dividend = (Mode32) ? ACPI_UINT32_MAX : ACPI_UINT64_MAX;

285     /* Main loop: convert the string to a 32- or 64-bit integer */

287     while (*String)
288     {
289         if (ACPI_IS_DIGIT (*String))
290         {
291             /* Convert ASCII 0-9 to Decimal value */

293             ThisDigit = ((UINT8) *String) - '0';
294         }
295         else if (Base == 10)
296         {
297             /* Digit is out of range; possible in ToInteger case only */

299             Term = 1;
300         }
301         else
302         {
303             ThisDigit = (UINT8) ACPI_TOUPPER (*String);
304             if (ACPI_IS_XDIGIT ((char) ThisDigit))
305             {
306                 /* Convert ASCII Hex char to value */

308                 ThisDigit = ThisDigit - 'A' + 10;
309             }
310             else
311             {
312                 Term = 1;
313             }
314         }

316         if (Term)
317         {
318             if (ToIntegerOp)
319             {
320                 goto ErrorExit;
321             }
322             else
323             {
324                 break;
```



```
325             }
326         }
327         else if ((ValidDigits == 0) && (ThisDigit == 0) && !SignOf0x)
328         {
329             /* Skip zeros */
330             String++;
331             continue;
332         }

334         ValidDigits++;

336         if (SignOf0x && ((ValidDigits > 16) || ((ValidDigits > 8) && Mode32)))
337         {
338             /*
339              * This is ToInteger operation case.
340              * No any restrictions for string-to-integer conversion,
341              * see ACPI spec.
342              */
343             goto ErrorExit;
344         }

346         /* Divide the digit into the correct position */

348         (void) AcpiUtShortDivide ((Dividend - (UINT64) ThisDigit),
349                     Base, &Quotient, NULL);

351         if (ReturnValue > Quotient)
352         {
353             if (ToIntegerOp)
354             {
355                 goto ErrorExit;
356             }
357             else
358             {
359                 break;
360             }
361         }

363         ReturnValue *= Base;
364         ReturnValue += ThisDigit;
365         String++;
366     }

368     /* All done, normal exit */

370 AllDone:

372     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Converted value: %8.8X%8.8X\n",
373         ACPI_FORMAT_UINT64 (ReturnValue)));

375     *RetInteger = ReturnValue;
376     return_ACPI_STATUS (AE_OK);


379 ErrorExit:
380     /* Base was set/validated above */

382     if (Base == 10)
383     {
384         return_ACPI_STATUS (AE_BAD_DECIMAL_CONSTANT);
385     }
386     else
387     {
388         return_ACPI_STATUS (AE_BAD_HEX_CONSTANT);
389     }
390 }
```

```
393 /*******************************************************************************
394  *
395  * FUNCTION:    AcpiUtPrintString
396  *
397  * PARAMETERS:  String          - Null terminated ASCII string
398  *              MaxLength       - Maximum output length. Used to constrain the
399  *                                length of strings during debug output only.
400  *
401  * RETURN:      None
402  *
403  * DESCRIPTION: Dump an ASCII string with support for ACPI-defined escape
404  *              sequences.
405  *
406  ******************************************************************************/

408 void
409 AcpiUtPrintString (
410     char                    *String,
411     UINT16                  MaxLength)
412 {
413     UINT32                  i;


416     if (!String)
417     {
418         AcpiOsPrintf ("<\"NULL STRING PTR\">");
419         return;
420     }

422     AcpiOsPrintf ("\"");
423     for (i = 0; String[i] && (i < MaxLength); i++)
424     {
425         /* Escape sequences */

427         switch (String[i])
428         {
429         case 0x07:

431             AcpiOsPrintf ("\\a");       /* BELL */
432             break;

434         case 0x08:

436             AcpiOsPrintf ("\\b");       /* BACKSPACE */
437             break;

439         case 0x0C:

441             AcpiOsPrintf ("\\f");       /* FORMFEED */
442             break;

444         case 0x0A:

446             AcpiOsPrintf ("\\n");       /* LINEFEED */
447             break;

449         case 0x0D:

451             AcpiOsPrintf ("\\r");       /* CARRIAGE RETURN*/
452             break;

454         case 0x09:

456             AcpiOsPrintf ("\\t");       /* HORIZONTAL TAB */
```



```
457             break;

459         case 0x0B:

461             AcpiOsPrintf ("\\v");       /* VERTICAL TAB */
462             break;

464         case '\'':                      /* Single Quote */
465         case '\"':                      /* Double Quote */
466         case '\\':                      /* Backslash */

468             AcpiOsPrintf ("\\%c", (int) String[i]);
469             break;

471         default:

473             /* Check for printable character or hex escape */

475             if (ACPI_IS_PRINT (String[i]))
476             {
477                 /* This is a normal character */

479                 AcpiOsPrintf ("%c", (int) String[i]);
480             }
481             else
482             {
483                 /* All others will be Hex escapes */

485                 AcpiOsPrintf ("\\x%2.2X", (INT32) String[i]);
486             }
487             break;
488         }
489     }
490     AcpiOsPrintf ("\"");

492     if (i == MaxLength && String[i])
493     {
494         AcpiOsPrintf ("...");
495     }
496 }


499 /*******************************************************************************
500  *
501  * FUNCTION:    AcpiUtValidAcpiChar
502  *
503  * PARAMETERS:  Char            - The character to be examined
504  *              Position        - Byte position (0-3)
505  *
506  * RETURN:      TRUE if the character is valid, FALSE otherwise
507  *
508  * DESCRIPTION: Check for a valid ACPI character. Must be one of:
509  *              1) Upper case alpha
510  *              2) numeric
511  *              3) underscore
512  *
513  *              We allow a '!' as the last character because of the ASF! table
514  *
515  ******************************************************************************/

517 BOOLEAN
518 AcpiUtValidAcpiChar (
519     char                    Character,
520     UINT32                  Position)
521 {
```

```
523     if (!((Character >= 'A' && Character <= 'Z') ||
524           (Character >= '0' && Character <= '9') ||
525           (Character == '_')))
526     {
527         /* Allow a '!' in the last position */
528
529         if (Character == '!' && Position == 3)
530         {
531             return (TRUE);
532         }
533
534         return (FALSE);
535     }
536
537     return (TRUE);
538 }
539
540
541 /*******************************************************************************
542  *
543  * FUNCTION:    AcpiUtValidAcpiName
544  *
545  * PARAMETERS:  Name            - The name to be examined. Does not have to
546  *                                be NULL terminated string.
547  *
548  * RETURN:      TRUE if the name is valid, FALSE otherwise
549  *
550  * DESCRIPTION: Check for a valid ACPI name. Each character must be one of:
551  *              1) Upper case alpha
552  *              2) numeric
553  *              3) underscore
554  *
555  ******************************************************************************/
556
557 BOOLEAN
558 AcpiUtValidAcpiName (
559     char                    *Name)
560 {
561     UINT32                  i;
562
563
564     ACPI_FUNCTION_ENTRY ();
565
566
567     for (i = 0; i < ACPI_NAME_SIZE; i++)
568     {
569         if (!AcpiUtValidAcpiChar (Name[i], i))
570         {
571             return (FALSE);
572         }
573     }
574
575     return (TRUE);
576 }
577
578
579 /*******************************************************************************
580  *
581  * FUNCTION:    AcpiUtRepairName
582  *
583  * PARAMETERS:  Name            - The ACPI name to be repaired
584  *
585  * RETURN:      Repaired version of the name
586  *
587  * DESCRIPTION: Repair an ACPI name: Change invalid characters to '*' and
588  *              return the new name. NOTE: the Name parameter must reside in
```



```
589  *              read/write memory, cannot be a const.
590  *
591  * An ACPI Name must consist of valid ACPI characters. We will repair the name
592  * if necessary because we don't want to abort because of this, but we want
593  * all namespace names to be printable. A warning message is appropriate.
594  *
595  * This issue came up because there are in fact machines that exhibit
596  * this problem, and we want to be able to enable ACPI support for them,
597  * even though there are a few bad names.
598  *
599  ******************************************************************************/
600
601 void
602 AcpiUtRepairName (
603     char                    *Name)
604 {
605     UINT32                  i;
606     BOOLEAN                 FoundBadChar = FALSE;
607     UINT32                  OriginalName;
608
609
610     ACPI_FUNCTION_NAME (UtRepairName);
611
612
613     ACPI_MOVE_NAME (&OriginalName, Name);
614
615     /* Check each character in the name */
616
617     for (i = 0; i < ACPI_NAME_SIZE; i++)
618     {
619         if (AcpiUtValidAcpiChar (Name[i], i))
620         {
621             continue;
622         }
623
624         /*
625          * Replace a bad character with something printable, yet technically
626          * still invalid. This prevents any collisions with existing "good"
627          * names in the namespace.
628          */
629         Name[i] = '*';
630         FoundBadChar = TRUE;
631     }
632
633     if (FoundBadChar)
634     {
635         /* Report warning only if in strict mode or debug mode */
636
637         if (!AcpiGbl_EnableInterpreterSlack)
638         {
639             ACPI_WARNING ((AE_INFO,
640                 "Invalid character(s) in name (0x%.8X), repaired: [%4.4s]",
641                 OriginalName, Name));
642         }
643         else
644         {
645             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
646                 "Invalid character(s) in name (0x%.8X), repaired: [%4.4s]",
647                 OriginalName, Name));
648         }
649     }
650 }
651
652
653 #if defined ACPI_ASL_COMPILER || defined ACPI_EXEC_APP
654 /*******************************************************************************
```

```
 *
 * FUNCTION:    UtConvertBackslashes
 *
 * PARAMETERS:  Pathname        - File pathname string to be converted
 *
 * RETURN:      Modifies the input Pathname
 *
 * DESCRIPTION: Convert all backslashes (0x5C) to forward slashes (0x2F) within
 *              the entire input file pathname string.
 *
 ******************************************************************************/

void
UtConvertBackslashes (
    char                    *Pathname)
{

    if (!Pathname)
    {
        return;
    }

    while (*Pathname)
    {
        if (*Pathname == '\\')
        {
            *Pathname = '/';
        }

        Pathname++;
    }
}
#endif

#if defined (ACPI_DEBUGGER) || defined (ACPI_APPLICATION)
/******************************************************************************
 *
 * FUNCTION:    AcpiUtSafeStrcpy, AcpiUtSafeStrcat, AcpiUtSafeStrncat
 *
 * PARAMETERS:  Adds a "DestSize" parameter to each of the standard string
 *              functions. This is the size of the Destination buffer.
 *
 * RETURN:      TRUE if the operation would overflow the destination buffer.
 *
 * DESCRIPTION: Safe versions of standard Clib string functions. Ensure that
 *              the result of the operation will not overflow the output string
 *              buffer.
 *
 * NOTE:        These functions are typically only helpful for processing
 *              user input and command lines. For most ACPICA code, the
 *              required buffer length is precisely calculated before buffer
 *              allocation, so the use of these functions is unnecessary.
 *
 ******************************************************************************/

BOOLEAN
AcpiUtSafeStrcpy (
    char                    *Dest,
    ACPI_SIZE               DestSize,
    char                    *Source)
{

    if (ACPI_STRLEN (Source) >= DestSize)
    {
        return (TRUE);
    }
```



```
    ACPI_STRCPY (Dest, Source);
    return (FALSE);
}

BOOLEAN
AcpiUtSafeStrcat (
    char                    *Dest,
    ACPI_SIZE               DestSize,
    char                    *Source)
{

    if ((ACPI_STRLEN (Dest) + ACPI_STRLEN (Source)) >= DestSize)
    {
        return (TRUE);
    }

    ACPI_STRCAT (Dest, Source);
    return (FALSE);
}

BOOLEAN
AcpiUtSafeStrncat (
    char                    *Dest,
    ACPI_SIZE               DestSize,
    char                    *Source,
    ACPI_SIZE               MaxTransferLength)
{
    ACPI_SIZE               ActualTransferLength;


    ActualTransferLength = ACPI_MIN (MaxTransferLength, ACPI_STRLEN (Source));

    if ((ACPI_STRLEN (Dest) + ActualTransferLength) >= DestSize)
    {
        return (TRUE);
    }

    ACPI_STRNCAT (Dest, Source, MaxTransferLength);
    return (FALSE);
}
#endif
```

```
**********************************************************
   23065 Thu Dec 26 13:49:44 2013
new/usr/src/common/acpica/components/utilities/uttrack.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: uttrack - Memory allocation tracking routines (debug only)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 /*
  45  * These procedures are used for tracking memory leaks in the subsystem, and
  46  * they get compiled out when the ACPI_DBG_TRACK_ALLOCATIONS is not set.
  47  *
  48  * Each memory allocation is tracked via a doubly linked list. Each
  49  * element contains the caller's component, module name, function name, and
  50  * line number. AcpiUtAllocate and AcpiUtAllocateZeroed call
  51  * AcpiUtTrackAllocation to add an element to the list; deletion
  52  * occurs in the body of AcpiUtFree.
  53  */

  55 #define __UTTRACK_C__

  57 #include "acpi.h"
  58 #include "accommon.h"
```



```
  60 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

  62 #define _COMPONENT          ACPI_UTILITIES
  63         ACPI_MODULE_NAME    ("uttrack")


  66 /* Local prototypes */

  68 static ACPI_DEBUG_MEM_BLOCK *
  69 AcpiUtFindAllocation (
  70     ACPI_DEBUG_MEM_BLOCK    *Allocation);
  69     void                    *Allocation);

  72 static ACPI_STATUS
  73 AcpiUtTrackAllocation (
  74     ACPI_DEBUG_MEM_BLOCK    *Address,
  75     ACPI_SIZE               Size,
  76     UINT8                   AllocType,
  77     UINT32                  Component,
  78     const char              *Module,
  79     UINT32                  Line);

  81 static ACPI_STATUS
  82 AcpiUtRemoveAllocation (
  83     ACPI_DEBUG_MEM_BLOCK    *Address,
  84     UINT32                  Component,
  85     const char              *Module,
  86     UINT32                  Line);


  89 /******************************************************************************
  90  *
  91  * FUNCTION:    AcpiUtCreateList
  92  *
  93  * PARAMETERS:  CacheName       - Ascii name for the cache
  94  *              ObjectSize      - Size of each cached object
  95  *              ReturnCache     - Where the new cache object is returned
  96  *
  97  * RETURN:      Status
  98  *
  99  * DESCRIPTION: Create a local memory list for tracking purposed
 100  *
 101  *****************************************************************************/

 103 ACPI_STATUS
 104 AcpiUtCreateList (
 105     char                    *ListName,
 106     UINT16                  ObjectSize,
 107     ACPI_MEMORY_LIST        **ReturnCache)
 108 {
 109     ACPI_MEMORY_LIST        *Cache;


 112     Cache = AcpiOsAllocate (sizeof (ACPI_MEMORY_LIST));
 113     if (!Cache)
 114     {
 115         return (AE_NO_MEMORY);
 116     }

 118     ACPI_MEMSET (Cache, 0, sizeof (ACPI_MEMORY_LIST));

 120     Cache->ListName   = ListName;
 121     Cache->ObjectSize = ObjectSize;

 123     *ReturnCache = Cache;
```

```
124     return (AE_OK);
125 }


128 /*******************************************************************************
129  *
130  * FUNCTION:    AcpiUtAllocateAndTrack
131  *
132  * PARAMETERS:  Size                - Size of the allocation
133  *              Component           - Component type of caller
134  *              Module              - Source file name of caller
135  *              Line                - Line number of caller
136  *
137  * RETURN:      Address of the allocated memory on success, NULL on failure.
138  *
139  * DESCRIPTION: The subsystem's equivalent of malloc.
140  *
141  ******************************************************************************/

143 void *
144 AcpiUtAllocateAndTrack (
145     ACPI_SIZE               Size,
146     UINT32                  Component,
147     const char              *Module,
148     UINT32                  Line)
149 {
150     ACPI_DEBUG_MEM_BLOCK    *Allocation;
151     ACPI_STATUS             Status;


154     /* Check for an inadvertent size of zero bytes */

156     if (!Size)
157     {
158         ACPI_WARNING ((Module, Line,
159             "Attempt to allocate zero bytes, allocating 1 byte"));
160         Size = 1;
161     }

163     Allocation = AcpiOsAllocate (Size + sizeof (ACPI_DEBUG_MEM_HEADER));
153     Allocation = AcpiUtAllocate (Size + sizeof (ACPI_DEBUG_MEM_HEADER),
154                     Component, Module, Line);
164     if (!Allocation)
165     {
166         /* Report allocation error */

168         ACPI_WARNING ((Module, Line,
169             "Could not allocate size %u", (UINT32) Size));

171         return (NULL);
172     }

174     Status = AcpiUtTrackAllocation (Allocation, Size,
175                     ACPI_MEM_MALLOC, Component, Module, Line);
176     if (ACPI_FAILURE (Status))
177     {
178         AcpiOsFree (Allocation);
179         return (NULL);
180     }

182     AcpiGbl_GlobalList->TotalAllocated++;
183     AcpiGbl_GlobalList->TotalSize += (UINT32) Size;
184     AcpiGbl_GlobalList->CurrentTotalSize += (UINT32) Size;
185     if (AcpiGbl_GlobalList->CurrentTotalSize > AcpiGbl_GlobalList->MaxOccupied)
186     {
187         AcpiGbl_GlobalList->MaxOccupied = AcpiGbl_GlobalList->CurrentTotalSize;
```



```
188     }

190     return ((void *) &Allocation->UserSpace);
191 }


194 /*******************************************************************************
195  *
196  * FUNCTION:    AcpiUtAllocateZeroedAndTrack
197  *
198  * PARAMETERS:  Size                - Size of the allocation
199  *              Component           - Component type of caller
200  *              Module              - Source file name of caller
201  *              Line                - Line number of caller
202  *
203  * RETURN:      Address of the allocated memory on success, NULL on failure.
204  *
205  * DESCRIPTION: Subsystem equivalent of calloc.
206  *
207  ******************************************************************************/

209 void *
210 AcpiUtAllocateZeroedAndTrack (
211     ACPI_SIZE               Size,
212     UINT32                  Component,
213     const char              *Module,
214     UINT32                  Line)
215 {
216     ACPI_DEBUG_MEM_BLOCK    *Allocation;
217     ACPI_STATUS             Status;


220     /* Check for an inadvertent size of zero bytes */

222     if (!Size)
223     {
224         ACPI_WARNING ((Module, Line,
225             "Attempt to allocate zero bytes, allocating 1 byte"));
226         Size = 1;
227     }

229     Allocation = AcpiOsAllocateZeroed (Size + sizeof (ACPI_DEBUG_MEM_HEADER));
206     Allocation = AcpiUtAllocateZeroed (Size + sizeof (ACPI_DEBUG_MEM_HEADER),
207                     Component, Module, Line);
230     if (!Allocation)
231     {
232         /* Report allocation error */

234         ACPI_ERROR ((Module, Line,
235             "Could not allocate size %u", (UINT32) Size));
236         return (NULL);
237     }

239     Status = AcpiUtTrackAllocation (Allocation, Size,
240                 ACPI_MEM_CALLOC, Component, Module, Line);
241     if (ACPI_FAILURE (Status))
242     {
243         AcpiOsFree (Allocation);
244         return (NULL);
245     }

247     AcpiGbl_GlobalList->TotalAllocated++;
248     AcpiGbl_GlobalList->TotalSize += (UINT32) Size;
249     AcpiGbl_GlobalList->CurrentTotalSize += (UINT32) Size;
250     if (AcpiGbl_GlobalList->CurrentTotalSize > AcpiGbl_GlobalList->MaxOccupied)
251     {
```

```
252         AcpiGbl_GlobalList->MaxOccupied = AcpiGbl_GlobalList->CurrentTotalSize;
253     }

255     return ((void *) &Allocation->UserSpace);
256 }
_____unchanged_portion_omitted_


315 /*******************************************************************************
316  *
317  * FUNCTION:    AcpiUtFindAllocation
318  *
319  * PARAMETERS:  Allocation              - Address of allocated memory
320  *
321  * RETURN:      Three cases:
322  *              1) List is empty, NULL is returned.
323  *              2) Element was found. Returns Allocation parameter.
324  *              3) Element was not found. Returns position where it should be
325  *                  inserted into the list.
299  * RETURN:      A list element if found; NULL otherwise.
326  *
327  * DESCRIPTION: Searches for an element in the global allocation tracking list.
328  *              If the element is not found, returns the location within the
329  *              list where the element should be inserted.
330  *
331  *              Note: The list is ordered by larger-to-smaller addresses.
332  *
333  *              This global list is used to detect memory leaks in ACPICA as
334  *              well as other issues such as an attempt to release the same
335  *              internal object more than once. Although expensive as far
336  *              as cpu time, this list is much more helpful for finding these
337  *              types of issues than using memory leak detectors outside of
338  *              the ACPICA code.
339  *
340  ******************************************************************************/

342 static ACPI_DEBUG_MEM_BLOCK *
343 AcpiUtFindAllocation (
344     ACPI_DEBUG_MEM_BLOCK    *Allocation)
307     void                    *Allocation)
345 {
346     ACPI_DEBUG_MEM_BLOCK    *Element;


312     ACPI_FUNCTION_ENTRY ();


349     Element = AcpiGbl_GlobalList->ListHead;
350     if (!Element)
351     {
352         return (NULL);
353     }

355     /*
356      * Search for the address.
357      *
358      * Note: List is ordered by larger-to-smaller addresses, on the
359      * assumption that a new allocation usually has a larger address
360      * than previous allocations.
361      */
362     while (Element > Allocation)
363     {
364         /* Check for end-of-list */
317     /* Search for the address. */

366         if (!Element->Next)
```



```
319     while (Element)
367         {
321         if (Element == Allocation)
322         {
368             return (Element);
369         }

371         Element = Element->Next;
372     }

374     if (Element == Allocation)
375     {
376         return (Element);
377     }

379     return (Element->Previous);
329     return (NULL);
380 }


383 /*******************************************************************************
384  *
385  * FUNCTION:    AcpiUtTrackAllocation
386  *
387  * PARAMETERS:  Allocation          - Address of allocated memory
388  *              Size                - Size of the allocation
389  *              AllocType           - MEM_MALLOC or MEM_CALLOC
390  *              Component           - Component type of caller
391  *              Module              - Source file name of caller
392  *              Line                - Line number of caller
393  *
394  * RETURN:      Status
344  * RETURN:      None.
395  *
396  * DESCRIPTION: Inserts an element into the global allocation tracking list.
397  *
398  ******************************************************************************/

400 static ACPI_STATUS
401 AcpiUtTrackAllocation (
402     ACPI_DEBUG_MEM_BLOCK    *Allocation,
403     ACPI_SIZE               Size,
404     UINT8                   AllocType,
405     UINT32                  Component,
406     const char              *Module,
407     UINT32                  Line)
408 {
409     ACPI_MEMORY_LIST        *MemList;
410     ACPI_DEBUG_MEM_BLOCK    *Element;
411     ACPI_STATUS             Status = AE_OK;


414     ACPI_FUNCTION_TRACE_PTR (UtTrackAllocation, Allocation);


417     if (AcpiGbl_DisableMemTracking)
418     {
419         return_ACPI_STATUS (AE_OK);
420     }

422     MemList = AcpiGbl_GlobalList;
423     Status = AcpiUtAcquireMutex (ACPI_MTX_MEMORY);
424     if (ACPI_FAILURE (Status))
425     {
426         return_ACPI_STATUS (Status);
427     }
```

```
    /*
     * Search the global list for this address to make sure it is not
     * already present. This will catch several kinds of problems.
     * Search list for this address to make sure it is not already on the list.
     * This will catch several kinds of problems.
     */
    Element = AcpiUtFindAllocation (Allocation);
    if (Element == Allocation)
    if (Element)
    {
        ACPI_ERROR ((AE_INFO,
            "UtTrackAllocation: Allocation (%p) already present in global list!"
            "UtTrackAllocation: Allocation already present in list! (%p)",
            Allocation));

        ACPI_ERROR ((AE_INFO, "Element %p Address %p",
            Element, Allocation));

        goto UnlockAndExit;
    }

    /* Fill in the instance data */
    /* Fill in the instance data. */

    Allocation->Size      = (UINT32) Size;
    Allocation->AllocType = AllocType;
    Allocation->Component = Component;
    Allocation->Line      = Line;

    ACPI_STRNCPY (Allocation->Module, Module, ACPI_MAX_MODULE_NAME);
    Allocation->Module[ACPI_MAX_MODULE_NAME-1] = 0;

    if (!Element)
    {
        /* Insert at list head */

        if (MemList->ListHead)
        {
            ((ACPI_DEBUG_MEM_BLOCK *)(MemList->ListHead))->Previous = Allocation
        }

        Allocation->Next = MemList->ListHead;
        Allocation->Previous = NULL;

        MemList->ListHead = Allocation;
    }
    else
    {
        /* Insert after element */

        Allocation->Next = Element->Next;
        Allocation->Previous = Element;

        if (Element->Next)
        {
            (Element->Next)->Previous = Allocation;
        }

        Element->Next = Allocation;
    }


UnlockAndExit:
    Status = AcpiUtReleaseMutex (ACPI_MTX_MEMORY);
    return_ACPI_STATUS (Status);
```



```
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiUtRemoveAllocation
 *
 * PARAMETERS:  Allocation          - Address of allocated memory
 *              Component           - Component type of caller
 *              Module              - Source file name of caller
 *              Line                - Line number of caller
 *
 * RETURN:      Status
 * RETURN:
 *
 * DESCRIPTION: Deletes an element from the global allocation tracking list.
 *
 ******************************************************************************/

static ACPI_STATUS
AcpiUtRemoveAllocation (
    ACPI_DEBUG_MEM_BLOCK    *Allocation,
    UINT32                  Component,
    const char              *Module,
    UINT32                  Line)
{
    ACPI_MEMORY_LIST        *MemList;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_NAME (UtRemoveAllocation);
    ACPI_FUNCTION_TRACE (UtRemoveAllocation);


    if (AcpiGbl_DisableMemTracking)
    {
        return (AE_OK);
        return_ACPI_STATUS (AE_OK);
    }

    MemList = AcpiGbl_GlobalList;
    if (NULL == MemList->ListHead)
    {
        /* No allocations! */

        ACPI_ERROR ((Module, Line,
            "Empty allocation list, nothing to free!"));

        return (AE_OK);
        return_ACPI_STATUS (AE_OK);
    }

    Status = AcpiUtAcquireMutex (ACPI_MTX_MEMORY);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
        return_ACPI_STATUS (Status);
    }

    /* Unlink */

    if (Allocation->Previous)
    {
        (Allocation->Previous)->Next = Allocation->Next;
    }
    else
```

```
546     {
547         MemList->ListHead = Allocation->Next;
548     }

550     if (Allocation->Next)
551     {
552         (Allocation->Next)->Previous = Allocation->Previous;
553     }

555     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Freeing %p, size 0%X\n",
556         &Allocation->UserSpace, Allocation->Size));

558     /* Mark the segment as deleted */

560     ACPI_MEMSET (&Allocation->UserSpace, 0xEA, Allocation->Size);

496     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Freeing size 0%X\n",
497         Allocation->Size));

562     Status = AcpiUtReleaseMutex (ACPI_MTX_MEMORY);
563     return (Status);
500     return_ACPI_STATUS (Status);
564 }


567 /******************************************************************************
568  *
569  * FUNCTION:    AcpiUtDumpAllocationInfo
570  *
571  * PARAMETERS:  None
508  * PARAMETERS:
572  *
573  * RETURN:      None
574  *
575  * DESCRIPTION: Print some info about the outstanding allocations.
576  *
577  *****************************************************************************/

579 void
580 AcpiUtDumpAllocationInfo (
581     void)
582 {
583 /*
584     ACPI_MEMORY_LIST        *MemList;
585 */

587     ACPI_FUNCTION_TRACE (UtDumpAllocationInfo);

589 /*
590     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
591                     ("%30s: %4d (%3d Kb)\n", "Current allocations",
592                     MemList->CurrentCount,
593                     ROUND_UP_TO_1K (MemList->CurrentSize)));

595     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
596                     ("%30s: %4d (%3d Kb)\n", "Max concurrent allocations",
597                     MemList->MaxConcurrentCount,
598                     ROUND_UP_TO_1K (MemList->MaxConcurrentSize)));


601     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
602                     ("%30s: %4d (%3d Kb)\n", "Total (all) internal objects",
603                     RunningObjectCount,
604                     ROUND_UP_TO_1K (RunningObjectSize)));

606     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
```



```
607                     ("%30s: %4d (%3d Kb)\n", "Total (all) allocations",
608                     RunningAllocCount,
609                     ROUND_UP_TO_1K (RunningAllocSize)));


612     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
613                     ("%30s: %4d (%3d Kb)\n", "Current Nodes",
614                     AcpiGbl_CurrentNodeCount,
615                     ROUND_UP_TO_1K (AcpiGbl_CurrentNodeSize)));

617     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
618                     ("%30s: %4d (%3d Kb)\n", "Max Nodes",
619                     AcpiGbl_MaxConcurrentNodeCount,
620                     ROUND_UP_TO_1K ((AcpiGbl_MaxConcurrentNodeCount *
621                         sizeof (ACPI_NAMESPACE_NODE)))));
622 */
623     return_VOID;
624 }


627 /******************************************************************************
628  *
629  * FUNCTION:    AcpiUtDumpAllocations
630  *
631  * PARAMETERS:  Component           - Component(s) to dump info for.
632  *              Module              - Module to dump info for. NULL means all.
633  *
634  * RETURN:      None
635  *
636  * DESCRIPTION: Print a list of all outstanding allocations.
637  *
638  *****************************************************************************/

640 void
641 AcpiUtDumpAllocations (
642     UINT32                  Component,
643     const char              *Module)
644 {
645     ACPI_DEBUG_MEM_BLOCK    *Element;
646     ACPI_DESCRIPTOR         *Descriptor;
647     UINT32                  NumOutstanding = 0;
648     UINT8                   DescriptorType;


651     ACPI_FUNCTION_TRACE (UtDumpAllocations);


654     if (AcpiGbl_DisableMemTracking)
655     {
656         return_VOID;
593         return;
657     }

659     /*
660      * Walk the allocation list.
661      */
662     if (ACPI_FAILURE (AcpiUtAcquireMutex (ACPI_MTX_MEMORY)))
663     {
664         return_VOID;
601         return;
665     }

667     Element = AcpiGbl_GlobalList->ListHead;
668     while (Element)
669     {
670         if ((Element->Component & Component) &&
```

```
671             ((Module == NULL) || (0 == ACPI_STRCMP (Module, Element->Module))))
672         {
673             Descriptor = ACPI_CAST_PTR (ACPI_DESCRIPTOR, &Element->UserSpace);

675             if (Element->Size < sizeof (ACPI_COMMON_DESCRIPTOR))
676             {
677                 AcpiOsPrintf ("%p Length 0x%04X %9.9s-%u "
678                     "[Not a Descriptor - too small]\n",
679                     Descriptor, Element->Size, Element->Module,
680                     Element->Line);
681             }
682             else
683             {
684                 /* Ignore allocated objects that are in a cache */

686                 if (ACPI_GET_DESCRIPTOR_TYPE (Descriptor) != ACPI_DESC_TYPE_CACH
687                 {
688                     AcpiOsPrintf ("%p Length 0x%04X %9.9s-%u [%s] ",
689                         Descriptor, Element->Size, Element->Module,
690                         Element->Line, AcpiUtGetDescriptorName (Descriptor));

692                     /* Validate the descriptor type using Type field and length

694                     DescriptorType = 0; /* Not a valid descriptor type */

696                     switch (ACPI_GET_DESCRIPTOR_TYPE (Descriptor))
697                     {
698                     case ACPI_DESC_TYPE_OPERAND:

700                         if (Element->Size == sizeof (ACPI_OPERAND_OBJECT))
636                         if (Element->Size == sizeof (ACPI_DESC_TYPE_OPERAND))
701                         {
702                             DescriptorType = ACPI_DESC_TYPE_OPERAND;
703                         }
704                         break;

706                     case ACPI_DESC_TYPE_PARSER:

708                         if (Element->Size == sizeof (ACPI_PARSE_OBJECT))
643                         if (Element->Size == sizeof (ACPI_DESC_TYPE_PARSER))
709                         {
710                             DescriptorType = ACPI_DESC_TYPE_PARSER;
711                         }
712                         break;

714                     case ACPI_DESC_TYPE_NAMED:

716                         if (Element->Size == sizeof (ACPI_NAMESPACE_NODE))
650                         if (Element->Size == sizeof (ACPI_DESC_TYPE_NAMED))
717                         {
718                             DescriptorType = ACPI_DESC_TYPE_NAMED;
719                         }
720                         break;

722                     default:

724                         break;
725                     }

727                     /* Display additional info for the major descriptor types */

729                     switch (DescriptorType)
730                     {
731                     case ACPI_DESC_TYPE_OPERAND:

733                         AcpiOsPrintf ("%12.12s  RefCount 0x%04X\n",
```



```
734                             AcpiUtGetTypeName (Descriptor->Object.Common.Type),
735                             Descriptor->Object.Common.ReferenceCount);
736                         break;

738                     case ACPI_DESC_TYPE_PARSER:

740                         AcpiOsPrintf ("AmlOpcode 0x%04hX\n",
741                             Descriptor->Op.Asl.AmlOpcode);
742                         break;

744                     case ACPI_DESC_TYPE_NAMED:

746                         AcpiOsPrintf ("%4.4s\n",
747                             AcpiUtGetNodeName (&Descriptor->Node));
748                         break;

750                     default:

752                         AcpiOsPrintf ( "\n");
753                         break;
754                     }
755                 }
756             }

758             NumOutstanding++;
759         }

761         Element = Element->Next;
762     }

764     (void) AcpiUtReleaseMutex (ACPI_MTX_MEMORY);

766     /* Print summary */

768     if (!NumOutstanding)
769     {
770         ACPI_INFO ((AE_INFO, "No outstanding allocations"));
771     }
772     else
773     {
774         ACPI_ERROR ((AE_INFO, "%u(0x%X) Outstanding allocations",
775             NumOutstanding, NumOutstanding));
776     }

778     return_VOID;
779 }

781 #endif  /* ACPI_DBG_TRACK_ALLOCATIONS */
```

```
**********************************************************
   18241 Thu Dec 26 13:49:44 2013
new/usr/src/common/acpica/components/utilities/utxface.c
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utxface - External interfaces, miscellaneous utility functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTXFACE_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acdebug.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utxface")


/*******************************************************************************
 *
 * FUNCTION:    AcpiTerminate
```



```
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Shutdown the ACPICA subsystem and release all resources.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiTerminate (
    void)
{
    ACPI_STATUS         Status;


    ACPI_FUNCTION_TRACE (AcpiTerminate);


    /* Just exit if subsystem is already shutdown */

    if (AcpiGbl_Shutdown)
    {
        ACPI_ERROR ((AE_INFO, "ACPI Subsystem is already terminated"));
        return_ACPI_STATUS (AE_OK);
    }

    /* Subsystem appears active, go ahead and shut it down */

    AcpiGbl_Shutdown = TRUE;
    AcpiGbl_StartupFlags = 0;
    ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Shutting down ACPI Subsystem\n"));

    /* Terminate the AML Debugger if present */

    ACPI_DEBUGGER_EXEC (AcpiGbl_DbTerminateThreads = TRUE);

    /* Shutdown and free all resources */

    AcpiUtSubsystemShutdown ();

    /* Free the mutex objects */

    AcpiUtMutexTerminate ();


#ifdef ACPI_DEBUGGER

    /* Shut down the debugger */

    AcpiDbTerminate ();
#endif

    /* Now we can shutdown the OS-dependent layer */

    Status = AcpiOsTerminate ();
    return_ACPI_STATUS (Status);
}

ACPI_EXPORT_SYMBOL_INIT (AcpiTerminate)


#ifndef ACPI_ASL_COMPILER
/*******************************************************************************
 *
 * FUNCTION:    AcpiSubsystemStatus
```

```
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status of the ACPI subsystem
 *
 * DESCRIPTION: Other drivers that use the ACPI subsystem should call this
 *              before making any other calls, to ensure the subsystem
 *              initialized successfully.
 *
 ****************************************************************************/

ACPI_STATUS
AcpiSubsystemStatus (
    void)
{

    if (AcpiGbl_StartupFlags & ACPI_INITIALIZED_OK)
    {
        return (AE_OK);
    }
    else
    {
        return (AE_ERROR);
    }
}

ACPI_EXPORT_SYMBOL (AcpiSubsystemStatus)


/*****************************************************************************
 *
 * FUNCTION:    AcpiGetSystemInfo
 *
 * PARAMETERS:  OutBuffer       - A buffer to receive the resources for the
 *                                device
 *
 * RETURN:      Status          - the status of the call
 *
 * DESCRIPTION: This function is called to get information about the current
 *              state of the ACPI subsystem. It will return system information
 *              in the OutBuffer.
 *
 *              If the function fails an appropriate status will be returned
 *              and the value of OutBuffer is undefined.
 *
 ****************************************************************************/

ACPI_STATUS
AcpiGetSystemInfo (
    ACPI_BUFFER             *OutBuffer)
{
    ACPI_SYSTEM_INFO        *InfoPtr;
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiGetSystemInfo);


    /* Parameter validation */

    Status = AcpiUtValidateBuffer (OutBuffer);
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }
```



```
    /* Validate/Allocate/Clear caller buffer */

    Status = AcpiUtInitializeBuffer (OutBuffer, sizeof (ACPI_SYSTEM_INFO));
    if (ACPI_FAILURE (Status))
    {
        return_ACPI_STATUS (Status);
    }

    /*
     * Populate the return buffer
     */
    InfoPtr = (ACPI_SYSTEM_INFO *) OutBuffer->Pointer;

    InfoPtr->AcpiCaVersion = ACPI_CA_VERSION;

    /* System flags (ACPI capabilities) */

    InfoPtr->Flags = ACPI_SYS_MODE_ACPI;

    /* Timer resolution - 24 or 32 bits  */

    if (AcpiGbl_FADT.Flags & ACPI_FADT_32BIT_TIMER)
    {
        InfoPtr->TimerResolution = 24;
    }
    else
    {
        InfoPtr->TimerResolution = 32;
    }

    /* Clear the reserved fields */

    InfoPtr->Reserved1 = 0;
    InfoPtr->Reserved2 = 0;

    /* Current debug levels */

    InfoPtr->DebugLayer = AcpiDbgLayer;
    InfoPtr->DebugLevel = AcpiDbgLevel;

    return_ACPI_STATUS (AE_OK);
}

ACPI_EXPORT_SYMBOL (AcpiGetSystemInfo)


/*****************************************************************************
 *
 * FUNCTION:    AcpiGetStatistics
 *
 * PARAMETERS:  Stats           - Where the statistics are returned
 *
 * RETURN:      Status          - the status of the call
 *
 * DESCRIPTION: Get the contents of the various system counters
 *
 *****************************************************************************/

ACPI_STATUS
AcpiGetStatistics (
    ACPI_STATISTICS         *Stats)
{
    ACPI_FUNCTION_TRACE (AcpiGetStatistics);


    /* Parameter validation */
```

```
258     if (!Stats)
259     {
260         return_ACPI_STATUS (AE_BAD_PARAMETER);
261     }

263     /* Various interrupt-based event counters */

265     Stats->SciCount = AcpiSciCount;
266     Stats->GpeCount = AcpiGpeCount;

268     ACPI_MEMCPY (Stats->FixedEventCount, AcpiFixedEventCount,
269         sizeof (AcpiFixedEventCount));


272     /* Other counters */

274     Stats->MethodCount = AcpiMethodCount;

276     return_ACPI_STATUS (AE_OK);
277 }

279 ACPI_EXPORT_SYMBOL (AcpiGetStatistics)


282 /*****************************************************************************
283  *
284  * FUNCTION:    AcpiInstallInitializationHandler
285  *
286  * PARAMETERS:  Handler             - Callback procedure
287  *              Function            - Not (currently) used, see below
288  *
289  * RETURN:      Status
290  *
291  * DESCRIPTION: Install an initialization handler
292  *
293  * TBD: When a second function is added, must save the Function also.
294  *
295  ****************************************************************************/

297 ACPI_STATUS
298 AcpiInstallInitializationHandler (
299     ACPI_INIT_HANDLER       Handler,
300     UINT32                  Function)
301 {

303     if (!Handler)
304     {
305         return (AE_BAD_PARAMETER);
306     }

308     if (AcpiGbl_InitHandler)
309     {
310         return (AE_ALREADY_EXISTS);
311     }

313     AcpiGbl_InitHandler = Handler;
314     return (AE_OK);
315 }

317 ACPI_EXPORT_SYMBOL (AcpiInstallInitializationHandler)


320 /*****************************************************************************
321  *
322  * FUNCTION:    AcpiPurgeCachedObjects
```



```
323  *
324  * PARAMETERS:  None
325  *
326  * RETURN:      Status
327  *
328  * DESCRIPTION: Empty all caches (delete the cached objects)
329  *
330  ****************************************************************************/

332 ACPI_STATUS
333 AcpiPurgeCachedObjects (
334     void)
335 {
336     ACPI_FUNCTION_TRACE (AcpiPurgeCachedObjects);


339     (void) AcpiOsPurgeCache (AcpiGbl_StateCache);
340     (void) AcpiOsPurgeCache (AcpiGbl_OperandCache);
341     (void) AcpiOsPurgeCache (AcpiGbl_PsNodeCache);
342     (void) AcpiOsPurgeCache (AcpiGbl_PsNodeExtCache);

344     return_ACPI_STATUS (AE_OK);
345 }

347 ACPI_EXPORT_SYMBOL (AcpiPurgeCachedObjects)


350 /*****************************************************************************
351  *
352  * FUNCTION:    AcpiInstallInterface
353  *
354  * PARAMETERS:  InterfaceName       - The interface to install
355  *
356  * RETURN:      Status
357  *
358  * DESCRIPTION: Install an _OSI interface to the global list
359  *
360  ****************************************************************************/

362 ACPI_STATUS
363 AcpiInstallInterface (
364     ACPI_STRING             InterfaceName)
365 {
366     ACPI_STATUS             Status;
367     ACPI_INTERFACE_INFO     *InterfaceInfo;


370     /* Parameter validation */

372     if (!InterfaceName || (ACPI_STRLEN (InterfaceName) == 0))
373     {
374         return (AE_BAD_PARAMETER);
375     }

377     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
378     if (ACPI_FAILURE (Status))
379     {
380         return (Status);
381     }

383     /* Check if the interface name is already in the global list */

385     InterfaceInfo = AcpiUtGetInterface (InterfaceName);
386     if (InterfaceInfo)
387     {
388         /*
```

```
389          * The interface already exists in the list. This is OK if the
390          * interface has been marked invalid -- just clear the bit.
391          */
392         if (InterfaceInfo->Flags & ACPI_OSI_INVALID)
393         {
394             InterfaceInfo->Flags &= ~ACPI_OSI_INVALID;
395             Status = AE_OK;
396         }
397         else
398         {
399             Status = AE_ALREADY_EXISTS;
400         }
401     }
402     else
403     {
404         /* New interface name, install into the global list */

406         Status = AcpiUtInstallInterface (InterfaceName);
407     }

409     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
410     return (Status);
411 }

413 ACPI_EXPORT_SYMBOL (AcpiInstallInterface)


416 /*******************************************************************************
417  *
418  * FUNCTION:    AcpiRemoveInterface
419  *
420  * PARAMETERS:  InterfaceName       - The interface to remove
421  *
422  * RETURN:      Status
423  *
424  * DESCRIPTION: Remove an _OSI interface from the global list
425  *
426  ******************************************************************************/

428 ACPI_STATUS
429 AcpiRemoveInterface (
430     ACPI_STRING             InterfaceName)
431 {
432     ACPI_STATUS             Status;


435     /* Parameter validation */

437     if (!InterfaceName || (ACPI_STRLEN (InterfaceName) == 0))
438     {
439         return (AE_BAD_PARAMETER);
440     }

442     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
443     if (ACPI_FAILURE (Status))
444     {
445         return (Status);
446     }

448     Status = AcpiUtRemoveInterface (InterfaceName);

450     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
451     return (Status);
452 }

454 ACPI_EXPORT_SYMBOL (AcpiRemoveInterface)
```



```
457 /*******************************************************************************
458  *
459  * FUNCTION:    AcpiInstallInterfaceHandler
460  *
461  * PARAMETERS:  Handler             - The _OSI interface handler to install
462  *                                    NULL means "remove existing handler"
463  *
464  * RETURN:      Status
465  *
466  * DESCRIPTION: Install a handler for the predefined _OSI ACPI method.
467  *              invoked during execution of the internal implementation of
468  *              _OSI. A NULL handler simply removes any existing handler.
469  *
470  ******************************************************************************/

472 ACPI_STATUS
473 AcpiInstallInterfaceHandler (
474     ACPI_INTERFACE_HANDLER  Handler)
475 {
476     ACPI_STATUS             Status;


479     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
480     if (ACPI_FAILURE (Status))
481     {
482         return (Status);
483     }

485     if (Handler && AcpiGbl_InterfaceHandler)
486     {
487         Status = AE_ALREADY_EXISTS;
488     }
489     else
490     {
491         AcpiGbl_InterfaceHandler = Handler;
492     }

494     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
495     return (Status);
496 }

498 ACPI_EXPORT_SYMBOL (AcpiInstallInterfaceHandler)


501 /*******************************************************************************
502  *
503  * FUNCTION:    AcpiUpdateInterfaces
504  *
505  * PARAMETERS:  Action              - Actions to be performed during the
506  *                                    update
507  *
508  * RETURN:      Status
509  *
510  * DESCRIPTION: Update _OSI interface strings, disabling or enabling OS vendor
511  *              string or/and feature group strings.
512  *
513  ******************************************************************************/

515 ACPI_STATUS
516 AcpiUpdateInterfaces (
517     UINT8                   Action)
518 {
519     ACPI_STATUS             Status;
```

```
522     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
523     if (ACPI_FAILURE (Status))
524     {
525         return (Status);
526     }

528     Status = AcpiUtUpdateInterfaces (Action);

530     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
531     return (Status);
532 }


535 /******************************************************************************
536  *
537  * FUNCTION:    AcpiCheckAddressRange
538  *
539  * PARAMETERS:  SpaceId             - Address space ID
540  *              Address             - Start address
541  *              Length              - Length
542  *              Warn                - TRUE if warning on overlap desired
543  *
544  * RETURN:      Count of the number of conflicts detected.
545  *
546  * DESCRIPTION: Check if the input address range overlaps any of the
547  *              ASL operation region address ranges.
548  *
549  *****************************************************************************/

551 UINT32
552 AcpiCheckAddressRange (
553     ACPI_ADR_SPACE_TYPE     SpaceId,
554     ACPI_PHYSICAL_ADDRESS   Address,
555     ACPI_SIZE               Length,
556     BOOLEAN                 Warn)
557 {
558     UINT32                  Overlaps;
559     ACPI_STATUS             Status;


562     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
563     if (ACPI_FAILURE (Status))
564     {
565         return (0);
566     }

568     Overlaps = AcpiUtCheckAddressRange (SpaceId, Address,
569         (UINT32) Length, Warn);

571     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
572     return (Overlaps);
573 }

575 ACPI_EXPORT_SYMBOL (AcpiCheckAddressRange)

577 #endif /* !ACPI_ASL_COMPILER */


580 /******************************************************************************
581  *
582  * FUNCTION:    AcpiDecodePldBuffer
583  *
584  * PARAMETERS:  InBuffer            - Buffer returned by _PLD method
585  *              Length              - Length of the InBuffer
586  *              ReturnBuffer        - Where the decode buffer is returned
```



```
587  *
588  * RETURN:      Status and the decoded _PLD buffer. User must deallocate
589  *              the buffer via ACPI_FREE.
590  *
591  * DESCRIPTION: Decode the bit-packed buffer returned by the _PLD method into
592  *              a local struct that is much more useful to an ACPI driver.
593  *
594  *****************************************************************************/

596 ACPI_STATUS
597 AcpiDecodePldBuffer (
598     UINT8                   *InBuffer,
599     ACPI_SIZE               Length,
600     ACPI_PLD_INFO           **ReturnBuffer)
601 {
602     ACPI_PLD_INFO           *PldInfo;
603     UINT32                  *Buffer = ACPI_CAST_PTR (UINT32, InBuffer);
604     UINT32                  Dword;


607     /* Parameter validation */

609     if (!InBuffer || !ReturnBuffer || (Length < 16))
610     {
611         return (AE_BAD_PARAMETER);
612     }

614     PldInfo = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_PLD_INFO));
615     if (!PldInfo)
616     {
617         return (AE_NO_MEMORY);
618     }

620     /* First 32-bit DWord */

622     ACPI_MOVE_32_TO_32 (&Dword, &Buffer[0]);
623     PldInfo->Revision =             ACPI_PLD_GET_REVISION (&Dword);
624     PldInfo->IgnoreColor =          ACPI_PLD_GET_IGNORE_COLOR (&Dword);
625     PldInfo->Color =                ACPI_PLD_GET_COLOR (&Dword);

627     /* Second 32-bit DWord */

629     ACPI_MOVE_32_TO_32 (&Dword, &Buffer[1]);
630     PldInfo->Width =                ACPI_PLD_GET_WIDTH (&Dword);
631     PldInfo->Height =               ACPI_PLD_GET_HEIGHT(&Dword);

633     /* Third 32-bit DWord */

635     ACPI_MOVE_32_TO_32 (&Dword, &Buffer[2]);
636     PldInfo->UserVisible =          ACPI_PLD_GET_USER_VISIBLE (&Dword);
637     PldInfo->Dock =                 ACPI_PLD_GET_DOCK (&Dword);
638     PldInfo->Lid =                  ACPI_PLD_GET_LID (&Dword);
639     PldInfo->Panel =                ACPI_PLD_GET_PANEL (&Dword);
640     PldInfo->VerticalPosition =     ACPI_PLD_GET_VERTICAL (&Dword);
641     PldInfo->HorizontalPosition =   ACPI_PLD_GET_HORIZONTAL (&Dword);
642     PldInfo->Shape =                ACPI_PLD_GET_SHAPE (&Dword);
643     PldInfo->GroupOrientation =     ACPI_PLD_GET_ORIENTATION (&Dword);
644     PldInfo->GroupToken =           ACPI_PLD_GET_TOKEN (&Dword);
645     PldInfo->GroupPosition =        ACPI_PLD_GET_POSITION (&Dword);
646     PldInfo->Bay =                  ACPI_PLD_GET_BAY (&Dword);

648     /* Fourth 32-bit DWord */

650     ACPI_MOVE_32_TO_32 (&Dword, &Buffer[3]);
651     PldInfo->Ejectable =            ACPI_PLD_GET_EJECTABLE (&Dword);
652     PldInfo->OspmEjectRequired =    ACPI_PLD_GET_OSPM_EJECT (&Dword);
```

```
653     PldInfo->CabinetNumber =        ACPI_PLD_GET_CABINET (&Dword);
654     PldInfo->CardCageNumber =       ACPI_PLD_GET_CARD_CAGE (&Dword);
655     PldInfo->Reference =            ACPI_PLD_GET_REFERENCE (&Dword);
656     PldInfo->Rotation =             ACPI_PLD_GET_ROTATION (&Dword);
657     PldInfo->Order =                ACPI_PLD_GET_ORDER (&Dword);

659     if (Length >= ACPI_PLD_BUFFER_SIZE)
660     {
661         /* Fifth 32-bit DWord (Revision 2 of _PLD) */

663         ACPI_MOVE_32_TO_32 (&Dword, &Buffer[4]);
664         PldInfo->VerticalOffset =       ACPI_PLD_GET_VERT_OFFSET (&Dword);
665         PldInfo->HorizontalOffset =     ACPI_PLD_GET_HORIZ_OFFSET (&Dword);
666     }

668     *ReturnBuffer = PldInfo;
669     return (AE_OK);
670 }

672 ACPI_EXPORT_SYMBOL (AcpiDecodePldBuffer)
```

```
**********************************************************
    8737 Thu Dec 26 13:49:45 2013
new/usr/src/common/acpica/components/utilities/utxferror.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utxferror - Various error/warning output functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTXFERROR_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utxferror")

/*
 * This module is used for the in-kernel ACPICA as well as the ACPICA
 * tools/applications.
 */

/******************************************************************************
```



```
 *
 * FUNCTION:    AcpiError
 *
 * PARAMETERS:  ModuleName          - Caller's module name (for error output)
 *              LineNumber          - Caller's line number (for error output)
 *              Format              - Printf format string + additional args
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print "ACPI Error" message with module/line/version info
 *
 *****************************************************************************/

void ACPI_INTERNAL_VAR_XFACE
AcpiError (
    const char              *ModuleName,
    UINT32                  LineNumber,
    const char              *Format,
    ...)
{
    va_list                 ArgList;


    ACPI_MSG_REDIRECT_BEGIN;
    AcpiOsPrintf (ACPI_MSG_ERROR);

    va_start (ArgList, Format);
    AcpiOsVprintf (Format, ArgList);
    ACPI_MSG_SUFFIX;
    va_end (ArgList);

    ACPI_MSG_REDIRECT_END;
}

ACPI_EXPORT_SYMBOL (AcpiError)


/******************************************************************************
 *
 * FUNCTION:    AcpiException
 *
 * PARAMETERS:  ModuleName          - Caller's module name (for error output)
 *              LineNumber          - Caller's line number (for error output)
 *              Status              - Status to be formatted
 *              Format              - Printf format string + additional args
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print "ACPI Exception" message with module/line/version info
 *              and decoded ACPI_STATUS.
 *
 *****************************************************************************/

void ACPI_INTERNAL_VAR_XFACE
AcpiException (
    const char              *ModuleName,
    UINT32                  LineNumber,
    ACPI_STATUS             Status,
    const char              *Format,
    ...)
{
    va_list                 ArgList;


    ACPI_MSG_REDIRECT_BEGIN;
    AcpiOsPrintf (ACPI_MSG_EXCEPTION "%s, ", AcpiFormatException (Status));
```

```
127     va_start (ArgList, Format);
128     AcpiOsVprintf (Format, ArgList);
129     ACPI_MSG_SUFFIX;
130     va_end (ArgList);
131
132     ACPI_MSG_REDIRECT_END;
133 }
134
135 ACPI_EXPORT_SYMBOL (AcpiException)
136
137
138 /*******************************************************************************
139  *
140  * FUNCTION:    AcpiWarning
141  *
142  * PARAMETERS:  ModuleName          - Caller's module name (for error output)
143  *              LineNumber          - Caller's line number (for error output)
144  *              Format              - Printf format string + additional args
145  *
146  * RETURN:      None
147  *
148  * DESCRIPTION: Print "ACPI Warning" message with module/line/version info
149  *
150  ******************************************************************************/
151
152 void ACPI_INTERNAL_VAR_XFACE
153 AcpiWarning (
154     const char              *ModuleName,
155     UINT32                  LineNumber,
156     const char              *Format,
157     ...)
158 {
159     va_list                 ArgList;
160
161
162     ACPI_MSG_REDIRECT_BEGIN;
163     AcpiOsPrintf (ACPI_MSG_WARNING);
164
165     va_start (ArgList, Format);
166     AcpiOsVprintf (Format, ArgList);
167     ACPI_MSG_SUFFIX;
168     va_end (ArgList);
169
170     ACPI_MSG_REDIRECT_END;
171 }
172
173 ACPI_EXPORT_SYMBOL (AcpiWarning)
174
175
176 /*******************************************************************************
177  *
178  * FUNCTION:    AcpiInfo
179  *
180  * PARAMETERS:  ModuleName          - Caller's module name (for error output)
181  *              LineNumber          - Caller's line number (for error output)
182  *              Format              - Printf format string + additional args
183  *
184  * RETURN:      None
185  *
186  * DESCRIPTION: Print generic "ACPI:" information message. There is no
187  *              module/line/version info in order to keep the message simple.
188  *
189  * TBD: ModuleName and LineNumber args are not needed, should be removed.
190  *
191  ******************************************************************************/
```



```
192
193 void ACPI_INTERNAL_VAR_XFACE
194 AcpiInfo (
195     const char              *ModuleName,
196     UINT32                  LineNumber,
197     const char              *Format,
198     ...)
199 {
200     va_list                 ArgList;
201
202
203     ACPI_MSG_REDIRECT_BEGIN;
204     AcpiOsPrintf (ACPI_MSG_INFO);
205
206     va_start (ArgList, Format);
207     AcpiOsVprintf (Format, ArgList);
208     AcpiOsPrintf ("\n");
209     va_end (ArgList);
210
211     ACPI_MSG_REDIRECT_END;
212 }
213
214 ACPI_EXPORT_SYMBOL (AcpiInfo)
215
216
217 /*******************************************************************************
218  *
219  * FUNCTION:    AcpiBiosError
220  *
221  * PARAMETERS:  ModuleName          - Caller's module name (for error output)
222  *              LineNumber          - Caller's line number (for error output)
223  *              Format              - Printf format string + additional args
224  *
225  * RETURN:      None
226  *
227  * DESCRIPTION: Print "ACPI Firmware Error" message with module/line/version
228  *              info
229  *
230  ******************************************************************************/
231
232 void ACPI_INTERNAL_VAR_XFACE
233 AcpiBiosError (
234     const char              *ModuleName,
235     UINT32                  LineNumber,
236     const char              *Format,
237     ...)
238 {
239     va_list                 ArgList;
240
241
242     ACPI_MSG_REDIRECT_BEGIN;
243     AcpiOsPrintf (ACPI_MSG_BIOS_ERROR);
244
245     va_start (ArgList, Format);
246     AcpiOsVprintf (Format, ArgList);
247     ACPI_MSG_SUFFIX;
248     va_end (ArgList);
249
250     ACPI_MSG_REDIRECT_END;
251 }
252
253 ACPI_EXPORT_SYMBOL (AcpiBiosError)
254
255
256 /*******************************************************************************
257  *
```

```
 * FUNCTION:    AcpiBiosWarning
 *
 * PARAMETERS:  ModuleName          - Caller's module name (for error output)
 *              LineNumber          - Caller's line number (for error output)
 *              Format              - Printf format string + additional args
 *
 * RETURN:      None
 *
 * DESCRIPTION: Print "ACPI Firmware Warning" message with module/line/version
 *              info
 *
 *****************************************************************************/

void ACPI_INTERNAL_VAR_XFACE
AcpiBiosWarning (
    const char              *ModuleName,
    UINT32                  LineNumber,
    const char              *Format,
    ...)
{
    va_list                 ArgList;


    ACPI_MSG_REDIRECT_BEGIN;
    AcpiOsPrintf (ACPI_MSG_BIOS_WARNING);

    va_start (ArgList, Format);
    AcpiOsVprintf (Format, ArgList);
    ACPI_MSG_SUFFIX;
    va_end (ArgList);

    ACPI_MSG_REDIRECT_END;
}

ACPI_EXPORT_SYMBOL (AcpiBiosWarning)
```

```
**********************************************************
   11038 Thu Dec 26 13:49:45 2013
new/usr/src/common/acpica/components/utilities/utxfinit.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utxfinit - External interfaces for ACPICA initialization
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define __UTXFINIT_C__
#define EXPORT_ACPI_INTERFACES

#include "acpi.h"
#include "accommon.h"
#include "acevents.h"
#include "acnamesp.h"
#include "acdebug.h"
#include "actables.h"

#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utxfinit")
```



```
/*******************************************************************************
 *
 * FUNCTION:    AcpiInitializeSubsystem
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Initializes all global variables. This is the first function
 *              called, so any early initialization belongs here.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiInitializeSubsystem (
    void)
{
    ACPI_STATUS             Status;


    ACPI_FUNCTION_TRACE (AcpiInitializeSubsystem);


    AcpiGbl_StartupFlags = ACPI_SUBSYSTEM_INITIALIZE;
    ACPI_DEBUG_EXEC (AcpiUtInitStackPtrTrace ());

    /* Initialize the OS-Dependent layer */

    Status = AcpiOsInitialize ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "During OSL initialization"));
        return_ACPI_STATUS (Status);
    }

    /* Initialize all globals used by the subsystem */

    Status = AcpiUtInitGlobals ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "During initialization of globals"));
        return_ACPI_STATUS (Status);
    }

    /* Create the default mutex objects */

    Status = AcpiUtMutexInitialize ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "During Global Mutex creation"));
        return_ACPI_STATUS (Status);
    }

    /*
     * Initialize the namespace manager and
     * the root of the namespace tree
     */
    Status = AcpiNsRootInitialize ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "During Namespace initialization"));
        return_ACPI_STATUS (Status);
    }

    /* Initialize the global OSI interfaces list with the static names */
```

```
125     Status = AcpiUtInitializeInterfaces ();
126     if (ACPI_FAILURE (Status))
127     {
128         ACPI_EXCEPTION ((AE_INFO, Status, "During OSI interfaces initialization"
129         return_ACPI_STATUS (Status);
130     }

132     /* If configured, initialize the AML debugger */

134 #ifdef ACPI_DEBUGGER
135     Status = AcpiDbInitialize ();
136     if (ACPI_FAILURE (Status))
137     {
138         ACPI_EXCEPTION ((AE_INFO, Status, "During Debugger initialization"));
139         return_ACPI_STATUS (Status);
140     }
141 #endif

143     return_ACPI_STATUS (AE_OK);
144 }

146 ACPI_EXPORT_SYMBOL_INIT (AcpiInitializeSubsystem)


149 /*******************************************************************************
150  *
151  * FUNCTION:    AcpiEnableSubsystem
152  *
153  * PARAMETERS:  Flags               - Init/enable Options
154  *
155  * RETURN:      Status
156  *
157  * DESCRIPTION: Completes the subsystem initialization including hardware.
158  *              Puts system into ACPI mode if it isn't already.
159  *
160  ******************************************************************************/

162 ACPI_STATUS
163 AcpiEnableSubsystem (
164     UINT32                  Flags)
165 {
166     ACPI_STATUS             Status = AE_OK;


169     ACPI_FUNCTION_TRACE (AcpiEnableSubsystem);


172 #if (!ACPI_REDUCED_HARDWARE)

174     /* Enable ACPI mode */

176     if (!(Flags & ACPI_NO_ACPI_ENABLE))
177     {
178         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "[Init] Going into ACPI mode\n"));

180         AcpiGbl_OriginalMode = AcpiHwGetMode();

182         Status = AcpiEnable ();
183         if (ACPI_FAILURE (Status))
184         {
185             ACPI_WARNING ((AE_INFO, "AcpiEnable failed"));
186             return_ACPI_STATUS (Status);
187         }
188     }

190     /*
```



```
191      * Obtain a permanent mapping for the FACS. This is required for the
192      * Global Lock and the Firmware Waking Vector
193      */
194     Status = AcpiTbInitializeFacs ();
195     if (ACPI_FAILURE (Status))
196     {
197         ACPI_WARNING ((AE_INFO, "Could not map the FACS table"));
198         return_ACPI_STATUS (Status);
199     }

201 #endif /* !ACPI_REDUCED_HARDWARE */

203     /*
204      * Install the default OpRegion handlers. These are installed unless
205      * other handlers have already been installed via the
206      * InstallAddressSpaceHandler interface.
207      */
208     if (!(Flags & ACPI_NO_ADDRESS_SPACE_INIT))
209     {
210         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
211             "[Init] Installing default address space handlers\n"));

213         Status = AcpiEvInstallRegionHandlers ();
214         if (ACPI_FAILURE (Status))
215         {
216             return_ACPI_STATUS (Status);
217         }
218     }

220 #if (!ACPI_REDUCED_HARDWARE)
221     /*
222      * Initialize ACPI Event handling (Fixed and General Purpose)
223      *
224      * Note1: We must have the hardware and events initialized before we can
225      * execute any control methods safely. Any control method can require
226      * ACPI hardware support, so the hardware must be fully initialized before
227      * any method execution!
228      *
229      * Note2: Fixed events are initialized and enabled here. GPEs are
230      * initialized, but cannot be enabled until after the hardware is
231      * completely initialized (SCI and GlobalLock activated) and the various
232      * initialization control methods are run (_REG, _STA, _INI) on the
233      * entire namespace.
234      */
235     if (!(Flags & ACPI_NO_EVENT_INIT))
236     {
237         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
238             "[Init] Initializing ACPI events\n"));

240         Status = AcpiEvInitializeEvents ();
241         if (ACPI_FAILURE (Status))
242         {
243             return_ACPI_STATUS (Status);
244         }
245     }

247     /*
248      * Install the SCI handler and Global Lock handler. This completes the
249      * hardware initialization.
250      */
251     if (!(Flags & ACPI_NO_HANDLER_INIT))
252     {
253         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
254             "[Init] Installing SCI/GL handlers\n"));

256         Status = AcpiEvInstallXruptHandlers ();
```

```
257         if (ACPI_FAILURE (Status))
258         {
259             return_ACPI_STATUS (Status);
260         }
261     }

263 #endif /* !ACPI_REDUCED_HARDWARE */

265     return_ACPI_STATUS (Status);
266 }

268 ACPI_EXPORT_SYMBOL_INIT (AcpiEnableSubsystem)


271 /*******************************************************************************
272  *
273  * FUNCTION:    AcpiInitializeObjects
274  *
275  * PARAMETERS:  Flags               - Init/enable Options
276  *
277  * RETURN:      Status
278  *
279  * DESCRIPTION: Completes namespace initialization by initializing device
280  *              objects and executing AML code for Regions, buffers, etc.
281  *
282  ******************************************************************************/

284 ACPI_STATUS
285 AcpiInitializeObjects (
286     UINT32                  Flags)
287 {
288     ACPI_STATUS             Status = AE_OK;


291     ACPI_FUNCTION_TRACE (AcpiInitializeObjects);


294     /*
295      * Run all _REG methods
296      *
297      * Note: Any objects accessed by the _REG methods will be automatically
298      * initialized, even if they contain executable AML (see the call to
299      * AcpiNsInitializeObjects below).
300      */
301     if (!(Flags & ACPI_NO_ADDRESS_SPACE_INIT))
302     {
303         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
304             "[Init] Executing _REG OpRegion methods\n"));

306         Status = AcpiEvInitializeOpRegions ();
307         if (ACPI_FAILURE (Status))
308         {
309             return_ACPI_STATUS (Status);
310         }
311     }

313     /*
314      * Execute any module-level code that was detected during the table load
315      * phase. Although illegal since ACPI 2.0, there are many machines that
316      * contain this type of code. Each block of detected executable AML code
317      * outside of any control method is wrapped with a temporary control
318      * method object and placed on a global list. The methods on this list
319      * are executed below.
320      */
321     AcpiNsExecModuleCodeList ();
```



```
323     /*
324      * Initialize the objects that remain uninitialized. This runs the
325      * executable AML that may be part of the declaration of these objects:
326      * OperationRegions, BufferFields, Buffers, and Packages.
327      */
328     if (!(Flags & ACPI_NO_OBJECT_INIT))
329     {
330         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
331             "[Init] Completing Initialization of ACPI Objects\n"));

333         Status = AcpiNsInitializeObjects ();
334         if (ACPI_FAILURE (Status))
335         {
336             return_ACPI_STATUS (Status);
337         }
338     }

340     /*
341      * Initialize all device objects in the namespace. This runs the device
342      * _STA and _INI methods.
343      */
344     if (!(Flags & ACPI_NO_DEVICE_INIT))
345     {
346         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
347             "[Init] Initializing ACPI Devices\n"));

349         Status = AcpiNsInitializeDevices ();
350         if (ACPI_FAILURE (Status))
351         {
352             return_ACPI_STATUS (Status);
353         }
354     }

356     /*
357      * Empty the caches (delete the cached objects) on the assumption that
358      * the table load filled them up more than they will be at runtime --
359      * thus wasting non-paged memory.
360      */
361     Status = AcpiPurgeCachedObjects ();

363     AcpiGbl_StartupFlags |= ACPI_INITIALIZED_OK;
364     return_ACPI_STATUS (Status);
365 }

367 ACPI_EXPORT_SYMBOL_INIT (AcpiInitializeObjects)
```

```
**********************************************************
    6663 Thu Dec 26 13:49:45 2013
new/usr/src/common/acpica/components/utilities/utxfmutex.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: utxfmutex - external AML mutex access functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __UTXFMUTEX_C__

#include "acpi.h"
#include "accommon.h"
#include "acnamesp.h"


#define _COMPONENT          ACPI_UTILITIES
        ACPI_MODULE_NAME    ("utxfmutex")


/* Local prototypes */

static ACPI_STATUS
AcpiUtGetMutexObject (
    ACPI_HANDLE             Handle,
    ACPI_STRING             Pathname,
    ACPI_OPERAND_OBJECT     **RetObj);
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiUtGetMutexObject
 *
 * PARAMETERS:  Handle              - Mutex or prefix handle (optional)
 *              Pathname            - Mutex pathname (optional)
 *              RetObj              - Where the mutex object is returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Get an AML mutex object. The mutex node is pointed to by
 *              Handle:Pathname. Either Handle or Pathname can be NULL, but
 *              not both.
 *
 *****************************************************************************/

static ACPI_STATUS
AcpiUtGetMutexObject (
    ACPI_HANDLE             Handle,
    ACPI_STRING             Pathname,
    ACPI_OPERAND_OBJECT     **RetObj)
{
    ACPI_NAMESPACE_NODE     *MutexNode;
    ACPI_OPERAND_OBJECT     *MutexObj;
    ACPI_STATUS             Status;


    /* Parameter validation */

    if (!RetObj || (!Handle && !Pathname))
    {
        return (AE_BAD_PARAMETER);
    }

    /* Get a the namespace node for the mutex */

    MutexNode = Handle;
    if (Pathname != NULL)
    {
        Status = AcpiGetHandle (Handle, Pathname,
            ACPI_CAST_PTR (ACPI_HANDLE, &MutexNode));
        if (ACPI_FAILURE (Status))
        {
            return (Status);
        }
    }

    /* Ensure that we actually have a Mutex object */

    if (!MutexNode ||
        (MutexNode->Type != ACPI_TYPE_MUTEX))
    {
        return (AE_TYPE);
    }

    /* Get the low-level mutex object */

    MutexObj = AcpiNsGetAttachedObject (MutexNode);
    if (!MutexObj)
    {
        return (AE_NULL_OBJECT);
    }

    *RetObj = MutexObj;
```

```
    return (AE_OK);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiAcquireMutex
 *
 * PARAMETERS:  Handle              - Mutex or prefix handle (optional)
 *              Pathname            - Mutex pathname (optional)
 *              Timeout             - Max time to wait for the lock (millisec)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Acquire an AML mutex. This is a device driver interface to
 *              AML mutex objects, and allows for transaction locking between
 *              drivers and AML code. The mutex node is pointed to by
 *              Handle:Pathname. Either Handle or Pathname can be NULL, but
 *              not both.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiAcquireMutex (
    ACPI_HANDLE             Handle,
    ACPI_STRING             Pathname,
    UINT16                  Timeout)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *MutexObj;


    /* Get the low-level mutex associated with Handle:Pathname */

    Status = AcpiUtGetMutexObject (Handle, Pathname, &MutexObj);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Acquire the OS mutex */

    Status = AcpiOsAcquireMutex (MutexObj->Mutex.OsMutex, Timeout);
    return (Status);
}


/*******************************************************************************
 *
 * FUNCTION:    AcpiReleaseMutex
 *
 * PARAMETERS:  Handle              - Mutex or prefix handle (optional)
 *              Pathname            - Mutex pathname (optional)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Release an AML mutex. This is a device driver interface to
 *              AML mutex objects, and allows for transaction locking between
 *              drivers and AML code. The mutex node is pointed to by
 *              Handle:Pathname. Either Handle or Pathname can be NULL, but
 *              not both.
 *
 ******************************************************************************/

ACPI_STATUS
AcpiReleaseMutex (
```



```
    ACPI_HANDLE             Handle,
    ACPI_STRING             Pathname)
{
    ACPI_STATUS             Status;
    ACPI_OPERAND_OBJECT     *MutexObj;


    /* Get the low-level mutex associated with Handle:Pathname */

    Status = AcpiUtGetMutexObject (Handle, Pathname, &MutexObj);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Release the OS mutex */

    AcpiOsReleaseMutex (MutexObj->Mutex.OsMutex);
    return (AE_OK);
}
```

```
**********************************************************
    6054 Thu Dec 26 13:49:45 2013
new/usr/src/common/acpica/include/acapps.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acapps - common include for ACPI applications/tools
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _ACAPPS
#define _ACAPPS


#ifdef _MSC_VER                 /* disable some level-4 warnings */
#pragma warning(disable:4100)   /* warning C4100: unreferenced formal parameter
#endif

/* Common info for tool signons */

#define ACPICA_NAME                 "Intel ACPI Component Architecture"
#define ACPICA_COPYRIGHT            "Copyright (c) 2000 - 2013 Intel Corporation
#define ACPICA_COPYRIGHT            "Copyright (c) 2000 - 2011 Intel Corporation

#if ACPI_MACHINE_WIDTH == 64
#define ACPI_WIDTH          "-64"
```



```
#elif ACPI_MACHINE_WIDTH == 32
#define ACPI_WIDTH          "-32"

#else
#error unknown ACPI_MACHINE_WIDTH
#define ACPI_WIDTH          "-??"

#endif

/* Macros for signons and file headers */

#define ACPI_COMMON_SIGNON(UtilityName) \
    "\n%s\n%s version %8.8X%s [%s]\n%s\n\n", \
    ACPICA_NAME, \
    UtilityName, ((UINT32) ACPI_CA_VERSION), ACPI_WIDTH, __DATE__, \
    ACPICA_COPYRIGHT

#define ACPI_COMMON_HEADER(UtilityName, Prefix) \
    "%s%s\n%s%s version %8.8X%s [%s]\n%s%s\n%s\n", \
    Prefix, ACPICA_NAME, \
    Prefix, UtilityName, ((UINT32) ACPI_CA_VERSION), ACPI_WIDTH, __DATE__, \
    Prefix, ACPICA_COPYRIGHT, \
    Prefix

/* Macros for usage messages */

#define ACPI_USAGE_HEADER(Usage) \
    printf ("Usage: %s\nOptions:\n", Usage);

#define ACPI_OPTION(Name, Description) \
    printf ("  %-18s%s\n", Name, Description);


#define FILE_SUFFIX_DISASSEMBLY     "dsl"
#define ACPI_TABLE_FILE_SUFFIX      ".dat"


/*
 * getopt
 */
int
AcpiGetopt(
    int                     argc,
    char                    **argv,
    char                    *opts);

int
AcpiGetoptArgument (
    int                     argc,
    char                    **argv);

extern int                  AcpiGbl_Optind;
extern int                  AcpiGbl_Opterr;
extern int                  AcpiGbl_SubOptChar;
extern char                 *AcpiGbl_Optarg;


#ifndef ACPI_DUMP_APP
/*
 * adisasm
 */
ACPI_STATUS
AdAmlDisassemble (
    BOOLEAN                 OutToFile,
    char                    *Filename,
```

```
125     char                    *Prefix,
126     char                    **OutFilename,
127     BOOLEAN                 GetAllTables);

129 void
130 AdPrintStatistics (
131     void);

133 ACPI_STATUS
134 AdFindDsdt(
135     UINT8                   **DsdtPtr,
136     UINT32                  *DsdtLength);

138 void
139 AdDumpTables (
140     void);

142 ACPI_STATUS
143 AdGetLocalTables (
144     char                    *Filename,
145     BOOLEAN                 GetAllTables);

147 ACPI_STATUS
148 AdParseTable (
149     ACPI_TABLE_HEADER       *Table,
150     ACPI_OWNER_ID           *OwnerId,
151     BOOLEAN                 LoadTable,
152     BOOLEAN                 External);

154 ACPI_STATUS
155 AdDisplayTables (
156     char                    *Filename,
157     ACPI_TABLE_HEADER       *Table);

159 ACPI_STATUS
160 AdDisplayStatistics (
161     void);


164 /*
165  * adwalk
166  */
167 void
168 AcpiDmCrossReferenceNamespace (
169     ACPI_PARSE_OBJECT       *ParseTreeRoot,
170     ACPI_NAMESPACE_NODE     *NamespaceRoot,
171     ACPI_OWNER_ID           OwnerId);

173 void
174 AcpiDmDumpTree (
175     ACPI_PARSE_OBJECT       *Origin);

177 void
178 AcpiDmFindOrphanMethods (
179     ACPI_PARSE_OBJECT       *Origin);

181 void
182 AcpiDmFinishNamespaceLoad (
183     ACPI_PARSE_OBJECT       *ParseTreeRoot,
184     ACPI_NAMESPACE_NODE     *NamespaceRoot,
185     ACPI_OWNER_ID           OwnerId);

187 void
188 AcpiDmConvertResourceIndexes (
189     ACPI_PARSE_OBJECT       *ParseTreeRoot,
190     ACPI_NAMESPACE_NODE     *NamespaceRoot);
```



```
193 /*
194  * adfile
195  */
196 ACPI_STATUS
197 AdInitialize (
198     void);

200 char *
201 FlGenerateFilename (
202     char                    *InputFilename,
203     char                    *Suffix);

205 ACPI_STATUS
206 FlSplitInputPathname (
207     char                    *InputPath,
208     char                    **OutDirectoryPath,
209     char                    **OutFilename);

211 char *
212 AdGenerateFilename (
213     char                    *Prefix,
214     char                    *TableId);

216 void
217 AdWriteTable (
218     ACPI_TABLE_HEADER       *Table,
219     UINT32                  Length,
220     char                    *TableName,
221     char                    *OemTableId);
222 #endif

224 #endif /* _ACAPPS */
```

```
**********************************************************
   11132 Thu Dec 26 13:49:45 2013
new/usr/src/common/acpica/include/acbuffer.h
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acbuffer.h - Support for buffers returned by ACPI predefined names
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACBUFFER_H__
#define __ACBUFFER_H__

/*
 * Contains buffer structures for these predefined names:
 * _FDE, _GRT, _GTM, _PLD, _SRT
 */

/*
 * Note: C bitfields are not used for this reason:
 *
 * "Bitfields are great and easy to read, but unfortunately the C language
 * does not specify the layout of bitfields in memory, which means they are
 * essentially useless for dealing with packed data in on-disk formats or
 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
 * this decision was a design error in C. Ritchie could have picked an order
 * and stuck with it." Norman Ramsey.
 * See http://stackoverflow.com/a/1053662/41661
```



```
 */


/* _FDE return value */

typedef struct acpi_fde_info
{
    UINT32              Floppy0;
    UINT32              Floppy1;
    UINT32              Floppy2;
    UINT32              Floppy3;
    UINT32              Tape;

} ACPI_FDE_INFO;

/*
 * _GRT return value
 * _SRT input value
 */
typedef struct acpi_grt_info
{
    UINT16              Year;
    UINT8               Month;
    UINT8               Day;
    UINT8               Hour;
    UINT8               Minute;
    UINT8               Second;
    UINT8               Valid;
    UINT16              Milliseconds;
    UINT16              Timezone;
    UINT8               Daylight;
    UINT8               Reserved[3];

} ACPI_GRT_INFO;

/* _GTM return value */

typedef struct acpi_gtm_info
{
    UINT32              PioSpeed0;
    UINT32              DmaSpeed0;
    UINT32              PioSpeed1;
    UINT32              DmaSpeed1;
    UINT32              Flags;

} ACPI_GTM_INFO;

/*
 * Formatted _PLD return value. The minimum size is a package containing
 * one buffer.
 * Revision 1: Buffer is 16 bytes (128 bits)
 * Revision 2: Buffer is 20 bytes (160 bits)
 *
 * Note: This structure is returned from the AcpiDecodePldBuffer
 * interface.
 */
typedef struct acpi_pld_info
{
    UINT8               Revision;
    UINT8               IgnoreColor;
    UINT32              Color;
    UINT16              Width;
    UINT16              Height;
    UINT8               UserVisible;
    UINT8               Dock;
    UINT8               Lid;
```

```
    UINT8               Panel;
    UINT8               VerticalPosition;
    UINT8               HorizontalPosition;
    UINT8               Shape;
    UINT8               GroupOrientation;
    UINT8               GroupToken;
    UINT8               GroupPosition;
    UINT8               Bay;
    UINT8               Ejectable;
    UINT8               OspmEjectRequired;
    UINT8               CabinetNumber;
    UINT8               CardCageNumber;
    UINT8               Reference;
    UINT8               Rotation;
    UINT8               Order;
    UINT8               Reserved;
    UINT16              VerticalOffset;
    UINT16              HorizontalOffset;

} ACPI_PLD_INFO;


/*
 * Macros to:
 *     1) Convert a _PLD buffer to internal ACPI_PLD_INFO format - ACPI_PLD_GET*
 *        (Used by AcpiDecodePldBuffer)
 *     2) Construct a _PLD buffer - ACPI_PLD_SET*
 *        (Intended for BIOS use only)
 */
#define ACPI_PLD_REV1_BUFFER_SIZE               16 /* For Revision 1 of the buff
#define ACPI_PLD_BUFFER_SIZE                    20 /* For Revision 2 of the buff

/* First 32-bit dword, bits 0:32 */

#define ACPI_PLD_GET_REVISION(dword)            ACPI_GET_BITS (dword, 0, ACPI_7B
#define ACPI_PLD_SET_REVISION(dword,value)      ACPI_SET_BITS (dword, 0, ACPI_7B

#define ACPI_PLD_GET_IGNORE_COLOR(dword)        ACPI_GET_BITS (dword, 7, ACPI_1B
#define ACPI_PLD_SET_IGNORE_COLOR(dword,value)  ACPI_SET_BITS (dword, 7, ACPI_1B

#define ACPI_PLD_GET_COLOR(dword)               ACPI_GET_BITS (dword, 8, ACPI_24
#define ACPI_PLD_SET_COLOR(dword,value)         ACPI_SET_BITS (dword, 8, ACPI_24

/* Second 32-bit dword, bits 33:63 */

#define ACPI_PLD_GET_WIDTH(dword)               ACPI_GET_BITS (dword, 0, ACPI_16
#define ACPI_PLD_SET_WIDTH(dword,value)         ACPI_SET_BITS (dword, 0, ACPI_16

#define ACPI_PLD_GET_HEIGHT(dword)              ACPI_GET_BITS (dword, 16, ACPI_1
#define ACPI_PLD_SET_HEIGHT(dword,value)        ACPI_SET_BITS (dword, 16, ACPI_1

/* Third 32-bit dword, bits 64:95 */

#define ACPI_PLD_GET_USER_VISIBLE(dword)        ACPI_GET_BITS (dword, 0, ACPI_1B
#define ACPI_PLD_SET_USER_VISIBLE(dword,value)  ACPI_SET_BITS (dword, 0, ACPI_1B

#define ACPI_PLD_GET_DOCK(dword)                ACPI_GET_BITS (dword, 1, ACPI_1B
#define ACPI_PLD_SET_DOCK(dword,value)          ACPI_SET_BITS (dword, 1, ACPI_1B

#define ACPI_PLD_GET_LID(dword)                 ACPI_GET_BITS (dword, 2, ACPI_1B
#define ACPI_PLD_SET_LID(dword,value)           ACPI_SET_BITS (dword, 2, ACPI_1B

#define ACPI_PLD_GET_PANEL(dword)               ACPI_GET_BITS (dword, 3, ACPI_3B
#define ACPI_PLD_SET_PANEL(dword,value)         ACPI_SET_BITS (dword, 3, ACPI_3B

#define ACPI_PLD_GET_VERTICAL(dword)            ACPI_GET_BITS (dword, 6, ACPI_2B
```



```
#define ACPI_PLD_SET_VERTICAL(dword,value)      ACPI_SET_BITS (dword, 6, ACPI_2B

#define ACPI_PLD_GET_HORIZONTAL(dword)          ACPI_GET_BITS (dword, 8, ACPI_2B
#define ACPI_PLD_SET_HORIZONTAL(dword,value)    ACPI_SET_BITS (dword, 8, ACPI_2B

#define ACPI_PLD_GET_SHAPE(dword)               ACPI_GET_BITS (dword, 10, ACPI_4
#define ACPI_PLD_SET_SHAPE(dword,value)         ACPI_SET_BITS (dword, 10, ACPI_4

#define ACPI_PLD_GET_ORIENTATION(dword)         ACPI_GET_BITS (dword, 14, ACPI_1
#define ACPI_PLD_SET_ORIENTATION(dword,value)   ACPI_SET_BITS (dword, 14, ACPI_1

#define ACPI_PLD_GET_TOKEN(dword)               ACPI_GET_BITS (dword, 15, ACPI_8
#define ACPI_PLD_SET_TOKEN(dword,value)         ACPI_SET_BITS (dword, 15, ACPI_8

#define ACPI_PLD_GET_POSITION(dword)            ACPI_GET_BITS (dword, 23, ACPI_8
#define ACPI_PLD_SET_POSITION(dword,value)      ACPI_SET_BITS (dword, 23, ACPI_8

#define ACPI_PLD_GET_BAY(dword)                 ACPI_GET_BITS (dword, 31, ACPI_1
#define ACPI_PLD_SET_BAY(dword,value)           ACPI_SET_BITS (dword, 31, ACPI_1

/* Fourth 32-bit dword, bits 96:127 */

#define ACPI_PLD_GET_EJECTABLE(dword)           ACPI_GET_BITS (dword, 0, ACPI_1B
#define ACPI_PLD_SET_EJECTABLE(dword,value)     ACPI_SET_BITS (dword, 0, ACPI_1B

#define ACPI_PLD_GET_OSPM_EJECT(dword)          ACPI_GET_BITS (dword, 1, ACPI_1B
#define ACPI_PLD_SET_OSPM_EJECT(dword,value)    ACPI_SET_BITS (dword, 1, ACPI_1B

#define ACPI_PLD_GET_CABINET(dword)             ACPI_GET_BITS (dword, 2, ACPI_8B
#define ACPI_PLD_SET_CABINET(dword,value)       ACPI_SET_BITS (dword, 2, ACPI_8B

#define ACPI_PLD_GET_CARD_CAGE(dword)           ACPI_GET_BITS (dword, 10, ACPI_8
#define ACPI_PLD_SET_CARD_CAGE(dword,value)     ACPI_SET_BITS (dword, 10, ACPI_8

#define ACPI_PLD_GET_REFERENCE(dword)           ACPI_GET_BITS (dword, 18, ACPI_1
#define ACPI_PLD_SET_REFERENCE(dword,value)     ACPI_SET_BITS (dword, 18, ACPI_1

#define ACPI_PLD_GET_ROTATION(dword)            ACPI_GET_BITS (dword, 19, ACPI_4
#define ACPI_PLD_SET_ROTATION(dword,value)      ACPI_SET_BITS (dword, 19, ACPI_4

#define ACPI_PLD_GET_ORDER(dword)               ACPI_GET_BITS (dword, 23, ACPI_5
#define ACPI_PLD_SET_ORDER(dword,value)         ACPI_SET_BITS (dword, 23, ACPI_5

/* Fifth 32-bit dword, bits 128:159 (Revision 2 of _PLD only) */

#define ACPI_PLD_GET_VERT_OFFSET(dword)         ACPI_GET_BITS (dword, 0, ACPI_16
#define ACPI_PLD_SET_VERT_OFFSET(dword,value)   ACPI_SET_BITS (dword, 0, ACPI_16

#define ACPI_PLD_GET_HORIZ_OFFSET(dword)        ACPI_GET_BITS (dword, 16, ACPI_1
#define ACPI_PLD_SET_HORIZ_OFFSET(dword,value)  ACPI_SET_BITS (dword, 16, ACPI_1


#endif /* ACBUFFER_H */
```

```
**********************************************************
    2898 Thu Dec 26 13:49:46 2013
new/usr/src/common/acpica/include/accommon.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: accommon.h - Common include files for generation of ACPICA source
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACCOMMON_H__
  45 #define __ACCOMMON_H__

  47 /*
  48  * Common set of includes for all ACPICA source files.
  49  * We put them here because we don't want to duplicate them
  50  * in the the source code again and again.
  51  *
  52  * Note: The order of these include files is important.
  53  */
  54 #include "acconfig.h"           /* Global configuration constants */
  55 #include "acmacros.h"           /* C macros */
  56 #include "aclocal.h"            /* Internal data types */
  57 #include "acobject.h"           /* ACPI internal object */
  58 #include "acstruct.h"           /* Common structures */
  59 #include "acglobal.h"           /* All global variables */
```



```
  60 #include "achware.h"            /* Hardware defines and interfaces */
  61 #include "acutils.h"            /* Utility interfaces */


  64 #endif /* __ACCOMMON_H__ */
```

```
**********************************************************
    8300 Thu Dec 26 13:49:46 2013
new/usr/src/common/acpica/include/acconfig.h
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acconfig.h - Global configuration constants
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _ACCONFIG_H
#define _ACCONFIG_H


/******************************************************************************
 *
 * Configuration options
 *
 *****************************************************************************/

/*
 * ACPI_DEBUG_OUTPUT    - This switch enables all the debug facilities of the
 *                        ACPI subsystem. This includes the DEBUG_PRINT output
 *                        statements. When disabled, all DEBUG_PRINT
```



```
 *                        statements are compiled out.
 *
 * ACPI_APPLICATION     - Use this switch if the subsystem is going to be run
 *                        at the application level.
 *
 */

/*
 * OS name, used for the _OS object. The _OS object is essentially obsolete,
 * but there is a large base of ASL/AML code in existing machines that check
 * for the string below. The use of this string usually guarantees that
 * the ASL will execute down the most tested code path. Also, there is some
 * code that will not execute the _OSI method unless _OS matches the string
 * below. Therefore, change this string at your own risk.
 */
#define ACPI_OS_NAME                    "Microsoft Windows NT"

/* Maximum objects in the various object caches */

#define ACPI_MAX_STATE_CACHE_DEPTH      96          /* State objects */
#define ACPI_MAX_PARSE_CACHE_DEPTH      96          /* Parse tree objects */
#define ACPI_MAX_EXTPARSE_CACHE_DEPTH   96          /* Parse tree objects */
#define ACPI_MAX_OBJECT_CACHE_DEPTH     96          /* Interpreter operand objec
#define ACPI_MAX_NAMESPACE_CACHE_DEPTH  96          /* Namespace objects */

/*
 * Should the subsystem abort the loading of an ACPI table if the
 * table checksum is incorrect?
 */
#ifndef ACPI_CHECKSUM_ABORT
#define ACPI_CHECKSUM_ABORT             FALSE
#endif

/*
 * Generate a version of ACPICA that only supports "reduced hardware"
 * platforms (as defined in ACPI 5.0). Set to TRUE to generate a specialized
 * version of ACPICA that ONLY supports the ACPI 5.0 "reduced hardware"
 * model. In other words, no ACPI hardware is supported.
 *
 * If TRUE, this means no support for the following:
 *      PM Event and Control registers
 *      SCI interrupt (and handler)
 *      Fixed Events
 *      General Purpose Events (GPEs)
 *      Global Lock
 *      ACPI PM timer
 *      FACS table (Waking vectors and Global Lock)
 */
#ifndef ACPI_REDUCED_HARDWARE
#define ACPI_REDUCED_HARDWARE           FALSE
#endif


/******************************************************************************
 *
 * Subsystem Constants
 *
 *****************************************************************************/

/* Version of ACPI supported */

#define ACPI_CA_SUPPORT_LEVEL           5
#define ACPI_CA_SUPPORT_LEVEL           3

/* Maximum count for a semaphore object */
```

```
#define ACPI_MAX_SEMAPHORE_COUNT        256

/* Maximum object reference count (detects object deletion issues) */

#define ACPI_MAX_REFERENCE_COUNT        0x800

/* Default page size for use in mapping memory for operation regions */

#define ACPI_DEFAULT_PAGE_SIZE          4096    /* Must be power of 2 */

/* OwnerId tracking. 8 entries allows for 255 OwnerIds */

#define ACPI_NUM_OWNERID_MASKS          8

/* Size of the root table array is increased by this increment */

#define ACPI_ROOT_TABLE_SIZE_INCREMENT  4

/* Maximum number of While() loop iterations before forced abort */

#define ACPI_MAX_LOOP_ITERATIONS        0xFFFF

/* Maximum sleep allowed via Sleep() operator */

#define ACPI_MAX_SLEEP                  2000    /* 2000 millisec == two seconds
#define ACPI_MAX_SLEEP                  20000   /* Two seconds */

/* Address Range lists are per-SpaceId (Memory and I/O only) */

#define ACPI_ADDRESS_RANGE_MAX          2


/******************************************************************************
 *
 * ACPI Specification constants (Do not change unless the specification changes)
 *
 *****************************************************************************/

/* Method info (in WALK_STATE), containing local variables and argumetns */

#define ACPI_METHOD_NUM_LOCALS          8
#define ACPI_METHOD_MAX_LOCAL           7

#define ACPI_METHOD_NUM_ARGS            7
#define ACPI_METHOD_MAX_ARG             6

/*
 * Operand Stack (in WALK_STATE), Must be large enough to contain METHOD_MAX_ARG
 */
#define ACPI_OBJ_NUM_OPERANDS           8
#define ACPI_OBJ_MAX_OPERAND            7

/* Number of elements in the Result Stack frame, can be an arbitrary value */

#define ACPI_RESULTS_FRAME_OBJ_NUM      8

/*
 * Maximal number of elements the Result Stack can contain,
 * it may be an arbitray value not exceeding the types of
 * ResultSize and ResultCount (now UINT8).
 */
#define ACPI_RESULTS_OBJ_NUM_MAX        255

/* Constants used in searching for the RSDP in low memory */

#define ACPI_EBDA_PTR_LOCATION          0x0000040E     /* Physical Address */
```



```
#define ACPI_EBDA_PTR_LENGTH            2
#define ACPI_EBDA_WINDOW_SIZE           1024
#define ACPI_HI_RSDP_WINDOW_BASE        0x000E0000     /* Physical Address */
#define ACPI_HI_RSDP_WINDOW_SIZE        0x00020000
#define ACPI_RSDP_SCAN_STEP             16

/* Operation regions */

#define ACPI_USER_REGION_BEGIN          0x80

/* Maximum SpaceIds for Operation Regions */

#define ACPI_MAX_ADDRESS_SPACE          255
#define ACPI_NUM_DEFAULT_SPACES         4

/* Array sizes. Used for range checking also */

#define ACPI_MAX_MATCH_OPCODE           5

/* RSDP checksums */

#define ACPI_RSDP_CHECKSUM_LENGTH       20
#define ACPI_RSDP_XCHECKSUM_LENGTH      36

/* SMBus, GSBus and IPMI bidirectional buffer size */
/* SMBus and IPMI bidirectional buffer size */

#define ACPI_SMBUS_BUFFER_SIZE          34
#define ACPI_GSBUS_BUFFER_SIZE          34
#define ACPI_IPMI_BUFFER_SIZE           66

/* _SxD and _SxW control methods */

#define ACPI_NUM_SxD_METHODS            4
#define ACPI_NUM_SxW_METHODS            5


/******************************************************************************
 *
 * ACPI AML Debugger
 *
 *****************************************************************************/

#define ACPI_DEBUGGER_MAX_ARGS          ACPI_METHOD_NUM_ARGS + 4 /* Max command
#define ACPI_DEBUGGER_MAX_ARGS          ACPI_METHOD_NUM_ARGS + 2 /* Max command
#define ACPI_DB_LINE_BUFFER_SIZE        512

#define ACPI_DEBUGGER_COMMAND_PROMPT    '-'
#define ACPI_DEBUGGER_EXECUTE_PROMPT    '%'


#endif /* _ACCONFIG_H */
```

```
**********************************************************
   10173 Thu Dec 26 13:49:46 2013
new/usr/src/common/acpica/include/acdebug.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acdebug.h - ACPI/AML debugger
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACDEBUG_H__
#define __ACDEBUG_H__


#define ACPI_DEBUG_BUFFER_SIZE  0x4000      /* 16K buffer for return objects */

typedef struct acpi_db_command_info
typedef struct CommandInfo
{
    char                    *Name;          /* Command Name */
    UINT8                   MinArgs;        /* Minimum arguments required */

} ACPI_DB_COMMAND_INFO;
} COMMAND_INFO;
```



```
typedef struct acpi_db_command_help
typedef struct ArgumentInfo
{
    UINT8                   LineCount;      /* Number of help lines */
    char                    *Invocation;    /* Command Invocation */
    char                    *Description;   /* Command Description */

} ACPI_DB_COMMAND_HELP;

typedef struct acpi_db_argument_info
{
    char                    *Name;          /* Argument Name */

} ACPI_DB_ARGUMENT_INFO;
} ARGUMENT_INFO;

typedef struct acpi_db_execute_walk
typedef struct acpi_execute_walk
{
    UINT32                  Count;
    UINT32                  MaxCount;

} ACPI_DB_EXECUTE_WALK;
} ACPI_EXECUTE_WALK;


#define PARAM_LIST(pl)                  pl
#define DBTEST_OUTPUT_LEVEL(lvl)        if (AcpiGbl_DbOpt_verbose)
#define VERBOSE_PRINT(fp)               DBTEST_OUTPUT_LEVEL(lvl) {\
                                            AcpiOsPrintf PARAM_LIST(fp);}

#define EX_NO_SINGLE_STEP               1
#define EX_SINGLE_STEP                  2


/*
 * dbxface - external debugger interfaces
 */
ACPI_STATUS
AcpiDbInitialize (
    void);

void
AcpiDbTerminate (
    void);

ACPI_STATUS
AcpiDbSingleStep (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op,
    UINT32                  OpType);


/*
 * dbcmds - debug commands and output routines
 */
ACPI_NAMESPACE_NODE *
AcpiDbConvertToNode (
    char                    *InString);

void
AcpiDbDisplayTableInfo (
    char                    *TableArg);

void
AcpiDbDisplayTemplate (
```

```
119     char                    *BufferArg);

121 void
122 AcpiDbUnloadAcpiTable (
123     char                    *Name);
111     char                    *TableArg,
112     char                    *InstanceArg);

125 void
126 AcpiDbSendNotify (
127     char                    *Name,
128     UINT32                  Value);

130 void
131 AcpiDbDisplayInterfaces (
132     char                    *ActionArg,
133     char                    *InterfaceNameArg);

135 ACPI_STATUS
136 AcpiDbSleep (
137     char                    *ObjectArg);

139 void
140 AcpiDbDisplayLocks (
141     void);

143 void
144 AcpiDbDisplayResources (
145     char                    *ObjectArg);

147 ACPI_HW_DEPENDENT_RETURN_VOID (
148 void
149 AcpiDbDisplayGpes (
150     void))
138     void);

152 void
153 AcpiDbDisplayHandlers (
154     void);

156 ACPI_HW_DEPENDENT_RETURN_VOID (
157 void
158 AcpiDbGenerateGpe (
159     char                    *GpeArg,
160     char                    *BlockArg))
147     char                    *BlockArg);

162 ACPI_HW_DEPENDENT_RETURN_VOID (
163 void
164 AcpiDbGenerateSci (
165     void))

167 /*
168  * dbconvert - miscellaneous conversion routines
169  */
170 ACPI_STATUS
171 AcpiDbHexCharToValue (
172     int                     HexChar,
173     UINT8                   *ReturnValue);

175 ACPI_STATUS
176 AcpiDbConvertToPackage (
177     char                    *String,
178     ACPI_OBJECT             *Object);

180 ACPI_STATUS
```



```
181 AcpiDbConvertToObject (
182     ACPI_OBJECT_TYPE        Type,
183     char                    *String,
184     ACPI_OBJECT             *Object);

186 UINT8 *
187 AcpiDbEncodePldBuffer (
188     ACPI_PLD_INFO           *PldInfo);

190 void
191 AcpiDbDumpPldBuffer (
192     ACPI_OBJECT             *ObjDesc);


195 /*
196  * dbmethod - control method commands
197  */
198 void
199 AcpiDbSetMethodBreakpoint (
200     char                    *Location,
201     ACPI_WALK_STATE         *WalkState,
202     ACPI_PARSE_OBJECT       *Op);

204 void
205 AcpiDbSetMethodCallBreakpoint (
206     ACPI_PARSE_OBJECT       *Op);

208 void
209 AcpiDbSetMethodData (
210     char                    *TypeArg,
211     char                    *IndexArg,
212     char                    *ValueArg);

214 ACPI_STATUS
215 AcpiDbDisassembleMethod (
216     char                    *Name);

218 void
219 AcpiDbDisassembleAml (
220     char                    *Statements,
221     ACPI_PARSE_OBJECT       *Op);

223 void
224 AcpiDbBatchExecute (
225     char                    *CountArg);


228 /*
229  * dbnames - namespace commands
230  */
231 void
232 AcpiDbSetScope (
233     char                    *Name);

235 void
236 AcpiDbDumpNamespace (
237     char                    *StartArg,
238     char                    *DepthArg);

240 void
241 AcpiDbDumpNamespacePaths (
242     void);

244 void
245 AcpiDbDumpNamespaceByOwner (
246     char                    *OwnerArg,
```

```
247     char                    *DepthArg);

249 ACPI_STATUS
250 AcpiDbFindNameInNamespace (
251     char                    *NameArg);

253 void
254 AcpiDbCheckPredefinedNames (
255     void);

257 ACPI_STATUS
258 AcpiDbDisplayObjects (
259     char                    *ObjTypeArg,
260     char                    *DisplayCountArg);

262 void
263 AcpiDbCheckIntegrity (
264     void);

266 void
267 AcpiDbFindReferences (
268     char                    *ObjectArg);

270 void
271 AcpiDbGetBusInfo (
272     void);


275 /*
276  * dbdisply - debug display commands
277  */
278 void
279 AcpiDbDisplayMethodInfo (
280     ACPI_PARSE_OBJECT       *Op);

282 void
283 AcpiDbDecodeAndDisplayObject (
284     char                    *Target,
285     char                    *OutputType);

287 void
288 AcpiDbDisplayResultObject (
289     ACPI_OPERAND_OBJECT     *ObjDesc,
290     ACPI_WALK_STATE         *WalkState);

292 ACPI_STATUS
293 AcpiDbDisplayAllMethods (
294     char                    *DisplayCountArg);

296 void
297 AcpiDbDisplayArguments (
298     void);

300 void
301 AcpiDbDisplayLocals (
302     void);

304 void
305 AcpiDbDisplayResults (
306     void);

308 void
309 AcpiDbDisplayCallingTree (
310     void);

312 void
```



```
313 AcpiDbDisplayObjectType (
314     char                    *ObjectArg);

316 void
317 AcpiDbDisplayArgumentObject (
318     ACPI_OPERAND_OBJECT     *ObjDesc,
319     ACPI_WALK_STATE         *WalkState);


322 /*
323  * dbexec - debugger control method execution
324  */
325 void
326 AcpiDbExecute (
327     char                    *Name,
328     char                    **Args,
329     ACPI_OBJECT_TYPE        *Types,
330     UINT32                  Flags);

332 void
333 AcpiDbCreateExecutionThreads (
334     char                    *NumThreadsArg,
335     char                    *NumLoopsArg,
336     char                    *MethodNameArg);

338 void
339 AcpiDbDeleteObjects (
340     UINT32                  Count,
341     ACPI_OBJECT             *Objects);

343 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
344 UINT32
345 AcpiDbGetCacheInfo (
346     ACPI_MEMORY_LIST        *Cache);
347 #endif


350 /*
351  * dbfileio - Debugger file I/O commands
352  */
353 ACPI_OBJECT_TYPE
354 AcpiDbMatchArgument (
355     char                    *UserArgument,
356     ACPI_DB_ARGUMENT_INFO   *Arguments);
302     ARGUMENT_INFO           *Arguments);

358 void
359 AcpiDbCloseDebugFile (
360     void);

362 void
363 AcpiDbOpenDebugFile (
364     char                    *Name);

366 ACPI_STATUS
367 AcpiDbLoadAcpiTable (
368     char                    *Filename);

370 ACPI_STATUS
371 AcpiDbGetTableFromFile (
372     char                    *Filename,
373     ACPI_TABLE_HEADER       **Table);

375 ACPI_STATUS
376 AcpiDbReadTableFromFile (
377     char                    *Filename,
```

```
378     ACPI_TABLE_HEADER       **Table);


381 /*
382  * dbhistry - debugger HISTORY command
383  */
384 void
385 AcpiDbAddToHistory (
386     char                    *CommandLine);

388 void
389 AcpiDbDisplayHistory (
390     void);

392 char *
393 AcpiDbGetFromHistory (
394     char                    *CommandNumArg);

396 char *
397 AcpiDbGetHistoryByIndex (
398     UINT32                  CommanddNum);


401 /*
402  * dbinput - user front-end to the AML debugger
403  */
404 ACPI_STATUS
405 AcpiDbCommandDispatch (
406     char                    *InputBuffer,
407     ACPI_WALK_STATE         *WalkState,
408     ACPI_PARSE_OBJECT       *Op);

410 void ACPI_SYSTEM_XFACE
411 AcpiDbExecuteThread (
412     void                    *Context);

414 ACPI_STATUS
415 AcpiDbUserCommands (
416     char                    Prompt,
417     ACPI_PARSE_OBJECT       *Op);

419 char *
420 AcpiDbGetNextToken (
421     char                    *String,
422     char                    **Next,
423     ACPI_OBJECT_TYPE        *ReturnType);


426 /*
427  * dbstats - Generation and display of ACPI table statistics
428  */
429 void
430 AcpiDbGenerateStatistics (
431     ACPI_PARSE_OBJECT       *Root,
432     BOOLEAN                 IsMethod);

434 ACPI_STATUS
435 AcpiDbDisplayStatistics (
436     char                    *TypeArg);


439 /*
440  * dbutils - AML debugger utilities
441  */
442 void
443 AcpiDbSetOutputDestination (
```



```
444     UINT32                  Where);

446 void
447 AcpiDbDumpExternalObject (
448     ACPI_OBJECT             *ObjDesc,
449     UINT32                  Level);

451 void
452 AcpiDbPrepNamestring (
453     char                    *Name);

455 ACPI_NAMESPACE_NODE *
456 AcpiDbLocalNsLookup (
457     char                    *Name);

459 void
460 AcpiDbUint32ToHexString (
402 AcpiDbUInt32ToHexString (
461     UINT32                  Value,
462     char                    *Buffer);

464 #endif  /* __ACDEBUG_H__ */
```

```
**********************************************************
   24447 Thu Dec 26 13:49:47 2013
new/usr/src/common/acpica/include/acdisasm.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acdisasm.h - AML disassembler
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACDISASM_H__
#define __ACDISASM_H__

#include "amlresrc.h"


#define BLOCK_NONE              0
#define BLOCK_PAREN             1
#define BLOCK_BRACE             2
#define BLOCK_COMMA_LIST        4
#define ACPI_DEFAULT_RESNAME    *(UINT32 *) "__RD"

/*
 * Raw table data header. Used by disassembler and data table compiler.
 * Do not change.
 */
```



```
#define ACPI_RAW_TABLE_DATA_HEADER      "Raw Table Data"


typedef const struct acpi_dmtable_info
{
    UINT8                       Opcode;
    UINT16                      Offset;
    char                        *Name;
    UINT8                       Flags;

} ACPI_DMTABLE_INFO;

/* Values for Flags field above */

#define DT_LENGTH                       0x01    /* Field is a subtable length */
#define DT_FLAG                         0x02    /* Field is a flag value */
#define DT_NON_ZERO                     0x04    /* Field must be non-zero */
#define DT_OPTIONAL                     0x08    /* Field is optional */
#define DT_DESCRIBES_OPTIONAL           0x10    /* Field describes an optional f
#define DT_COUNT                        0x20    /* Currently not used */

/*
 * Values for Opcode above.
 * Note: 0-7 must not change, they are used as a flag shift value. Other
 * than those, new values can be added wherever appropriate.
 */
typedef enum
{
    /* Simple Data Types */

    ACPI_DMT_FLAG0          = 0,
    ACPI_DMT_FLAG1          = 1,
    ACPI_DMT_FLAG2          = 2,
    ACPI_DMT_FLAG3          = 3,
    ACPI_DMT_FLAG4          = 4,
    ACPI_DMT_FLAG5          = 5,
    ACPI_DMT_FLAG6          = 6,
    ACPI_DMT_FLAG7          = 7,
    ACPI_DMT_FLAGS0,
    ACPI_DMT_FLAGS1,
    ACPI_DMT_FLAGS2,
    ACPI_DMT_FLAGS4,
    ACPI_DMT_UINT8,
    ACPI_DMT_UINT16,
    ACPI_DMT_UINT24,
    ACPI_DMT_UINT32,
    ACPI_DMT_UINT40,
    ACPI_DMT_UINT48,
    ACPI_DMT_UINT56,
    ACPI_DMT_UINT64,
    ACPI_DMT_BUF7,
    ACPI_DMT_BUF10,
    ACPI_DMT_BUF16,
    ACPI_DMT_BUF128,
    ACPI_DMT_SIG,
    ACPI_DMT_STRING,
    ACPI_DMT_NAME4,
    ACPI_DMT_NAME6,
    ACPI_DMT_NAME8,

    /* Types that are decoded to strings and miscellaneous */

    ACPI_DMT_ACCWIDTH,
    ACPI_DMT_CHKSUM,
    ACPI_DMT_GAS,
    ACPI_DMT_SPACEID,
```

```
126     ACPI_DMT_UNICODE,
127     ACPI_DMT_UUID,

129     /* Types used only for the Data Table Compiler */

131     ACPI_DMT_BUFFER,
132     ACPI_DMT_DEVICE_PATH,
133     ACPI_DMT_LABEL,
134     ACPI_DMT_PCI_PATH,

136     /* Types that are specific to particular ACPI tables */

138     ACPI_DMT_ASF,
139     ACPI_DMT_DMAR,
140     ACPI_DMT_EINJACT,
141     ACPI_DMT_EINJINST,
142     ACPI_DMT_ERSTACT,
143     ACPI_DMT_ERSTINST,
144     ACPI_DMT_FADTPM,
145     ACPI_DMT_HEST,
146     ACPI_DMT_HESTNTFY,
147     ACPI_DMT_HESTNTYP,
148     ACPI_DMT_IVRS,
149     ACPI_DMT_MADT,
150     ACPI_DMT_PCCT,
151     ACPI_DMT_PMTT,
152     ACPI_DMT_SLIC,
153     ACPI_DMT_SRAT,

155     /* Special opcodes */

157     ACPI_DMT_EXTRA_TEXT,
158     ACPI_DMT_EXIT

160 } ACPI_ENTRY_TYPES;

162 typedef
163 void (*ACPI_DMTABLE_HANDLER) (
164     ACPI_TABLE_HEADER       *Table);

166 typedef
167 ACPI_STATUS (*ACPI_CMTABLE_HANDLER) (
168     void                    **PFieldList);

170 typedef struct acpi_dmtable_data
171 {
172     char                    *Signature;
173     ACPI_DMTABLE_INFO       *TableInfo;
174     ACPI_DMTABLE_HANDLER    TableHandler;
175     ACPI_CMTABLE_HANDLER    CmTableHandler;
176     const unsigned char     *Template;
177     char                    *Name;

179 } ACPI_DMTABLE_DATA;


182 typedef struct acpi_op_walk_info
183 {
184     UINT32                  Level;
185     UINT32                  LastLevel;
186     UINT32                  Count;
187     UINT32                  BitOffset;
188     UINT32                  Flags;
189     ACPI_WALK_STATE         *WalkState;

191 } ACPI_OP_WALK_INFO;
```



```
193 /*
194  * TBD - another copy of this is in asltypes.h, fix
195  */
196 #ifndef ASL_WALK_CALLBACK_DEFINED
197 typedef
198 ACPI_STATUS (*ASL_WALK_CALLBACK) (
199     ACPI_PARSE_OBJECT           *Op,
200     UINT32                      Level,
201     void                        *Context);
202 #define ASL_WALK_CALLBACK_DEFINED
203 #endif

205 typedef
206 void (*ACPI_RESOURCE_HANDLER) (
207     AML_RESOURCE            *Resource,
208     UINT32                  Length,
209     UINT32                  Level);

211 typedef struct acpi_resource_tag
212 {
213     UINT32                  BitIndex;
214     char                    *Tag;

216 } ACPI_RESOURCE_TAG;

218 /* Strings used for decoding flags to ASL keywords */

220 extern const char               *AcpiGbl_WordDecode[];
221 extern const char               *AcpiGbl_IrqDecode[];
222 extern const char               *AcpiGbl_LockRule[];
223 extern const char               *AcpiGbl_AccessTypes[];
224 extern const char               *AcpiGbl_UpdateRules[];
225 extern const char               *AcpiGbl_MatchOps[];

227 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf0[];
228 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf1[];
229 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf1a[];
230 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf2[];
231 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf2a[];
232 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf3[];
233 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsf4[];
234 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoAsfHdr[];
235 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoBoot[];
236 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoBert[];
237 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoBgrt[];
238 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoCpep[];
239 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoCpep0[];
240 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoCsrt0[];
241 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoCsrt1[];
242 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoCsrt2[];
243 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2[];
244 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2Device[];
245 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2Addr[];
246 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2Size[];
247 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2Name[];
248 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbg2OemData[];
249 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDbgp[];
250 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmar[];
251 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmarHdr[];
252 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmarScope[];
253 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmar0[];
254 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmar1[];
255 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmar2[];
256 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDmar3[];
257 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoDrtm[];
```

```
258 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoEcdt[];
259 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoEinj[];
260 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoEinj0[];
261 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoErst[];
262 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoErst0[];
263 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFacs[];
264 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFadt1[];
265 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFadt2[];
266 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFadt3[];
267 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFadt5[];
268 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFpdt[];
269 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFpdtHdr[];
270 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFpdt0[];
271 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoFpdt1[];
272 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoGas[];
273 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoGtdt[];
274 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHeader[];
275 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest[];
276 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest0[];
277 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest1[];
278 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest2[];
279 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest6[];
280 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest7[];
281 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest8[];
282 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHest9[];
283 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHestNotify[];
284 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHestBank[];
285 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoHpet[];
286 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs[];
287 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs0[];
288 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs1[];
289 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs4[];
290 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs8a[];
291 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs8b[];
292 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrs8c[];
293 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoIvrsHdr[];
294 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt[];
295 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt0[];
296 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt1[];
297 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt2[];
298 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt3[];
299 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt4[];
300 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt5[];
301 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt6[];
302 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt7[];
303 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt8[];
304 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt9[];
305 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt10[];
306 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt11[];
307 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadt12[];
308 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMadtHdr[];
309 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMcfg[];
310 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMcfg0[];
311 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMchi[];
312 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst[];
313 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst0[];
314 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst0A[];
315 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst0B[];
316 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst1[];
317 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMpst2[];
318 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMsct[];
319 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMsct0[];
320 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMtmr[];
321 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoMtmr0[];
322 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmtt[];
323 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmtt0[];
```



```
324 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmtt1[];
325 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmtt1a[];
326 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmtt2[];
327 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPmttHdr[];
328 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPcct[];
329 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPcctHdr[];
330 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoPcct0[];
331 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoRsdp1[];
332 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoRsdp2[];
333 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoS3pt[];
334 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoS3ptHdr[];
335 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoS3pt0[];
336 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoS3pt1[];
337 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSbst[];
338 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSlicHdr[];
339 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSlic0[];
340 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSlic1[];
341 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSlit[];
342 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSpcr[];
343 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSpmi[];
344 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSrat[];
345 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSratHdr[];
346 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSrat0[];
347 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSrat1[];
348 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoSrat2[];
349 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoTcpa[];
350 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoTpm2[];
351 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoUefi[];
352 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoVrtc[];
353 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoVrtc0[];
354 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoWaet[];
355 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoWdat[];
356 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoWdat0[];
357 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoWddt[];
358 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoWdrt[];

360 extern ACPI_DMTABLE_INFO        AcpiDmTableInfoGeneric[][2];


363 /*
364  * dmtable
365  */
366 extern ACPI_DMTABLE_DATA        AcpiDmTableData[];

368 UINT8
369 AcpiDmGenerateChecksum (
370     void                    *Table,
371     UINT32                  Length,
372     UINT8                   OriginalChecksum);

374 ACPI_DMTABLE_DATA *
375 AcpiDmGetTableData (
376     char                    *Signature);

378 void
379 AcpiDmDumpDataTable (
380     ACPI_TABLE_HEADER       *Table);

382 ACPI_STATUS
383 AcpiDmDumpTable (
384     UINT32                  TableLength,
385     UINT32                  TableOffset,
386     void                    *Table,
387     UINT32                  SubTableLength,
388     ACPI_DMTABLE_INFO       *Info);
```

```
390 void
391 AcpiDmLineHeader (
392     UINT32                  Offset,
393     UINT32                  ByteLength,
394     char                    *Name);

396 void
397 AcpiDmLineHeader2 (
398     UINT32                  Offset,
399     UINT32                  ByteLength,
400     char                    *Name,
401     UINT32                  Value);


404 /*
405  * dmtbdump
406  */
407 void
408 AcpiDmDumpAsf (
409     ACPI_TABLE_HEADER       *Table);

411 void
412 AcpiDmDumpCpep (
413     ACPI_TABLE_HEADER       *Table);

415 void
416 AcpiDmDumpCsrt (
417     ACPI_TABLE_HEADER       *Table);

419 void
420 AcpiDmDumpDbg2 (
421     ACPI_TABLE_HEADER       *Table);

423 void
424 AcpiDmDumpDmar (
425     ACPI_TABLE_HEADER       *Table);

427 void
428 AcpiDmDumpEinj (
429     ACPI_TABLE_HEADER       *Table);

431 void
432 AcpiDmDumpErst (
433     ACPI_TABLE_HEADER       *Table);

435 void
436 AcpiDmDumpFadt (
437     ACPI_TABLE_HEADER       *Table);

439 void
440 AcpiDmDumpFpdt (
441     ACPI_TABLE_HEADER       *Table);

443 void
444 AcpiDmDumpHest (
445     ACPI_TABLE_HEADER       *Table);

447 void
448 AcpiDmDumpIvrs (
449     ACPI_TABLE_HEADER       *Table);

451 void
452 AcpiDmDumpMadt (
453     ACPI_TABLE_HEADER       *Table);

455 void
```



```
456 AcpiDmDumpMcfg (
457     ACPI_TABLE_HEADER       *Table);

459 void
460 AcpiDmDumpMpst (
461     ACPI_TABLE_HEADER       *Table);

463 void
464 AcpiDmDumpMsct (
465     ACPI_TABLE_HEADER       *Table);

467 void
468 AcpiDmDumpMtmr (
469     ACPI_TABLE_HEADER       *Table);

471 void
472 AcpiDmDumpPcct (
473     ACPI_TABLE_HEADER       *Table);

475 void
476 AcpiDmDumpPmtt (
477     ACPI_TABLE_HEADER       *Table);

479 UINT32
480 AcpiDmDumpRsdp (
481     ACPI_TABLE_HEADER       *Table);

483 void
484 AcpiDmDumpRsdt (
485     ACPI_TABLE_HEADER       *Table);

487 UINT32
488 AcpiDmDumpS3pt (
489     ACPI_TABLE_HEADER       *Table);

491 void
492 AcpiDmDumpSlic (
493     ACPI_TABLE_HEADER       *Table);

495 void
496 AcpiDmDumpSlit (
497     ACPI_TABLE_HEADER       *Table);

499 void
500 AcpiDmDumpSrat (
501     ACPI_TABLE_HEADER       *Table);

503 void
504 AcpiDmDumpVrtc (
505     ACPI_TABLE_HEADER       *Table);

507 void
508 AcpiDmDumpWdat (
509     ACPI_TABLE_HEADER       *Table);

511 void
512 AcpiDmDumpXsdt (
513     ACPI_TABLE_HEADER       *Table);


516 /*
517  * dmwalk
518  */
519 void
520 AcpiDmDisassemble (
521     ACPI_WALK_STATE         *WalkState,
```

```
522     ACPI_PARSE_OBJECT       *Origin,
523     UINT32                  NumOpcodes);

525 void
526 AcpiDmWalkParseTree (
527     ACPI_PARSE_OBJECT       *Op,
528     ASL_WALK_CALLBACK       DescendingCallback,
529     ASL_WALK_CALLBACK       AscendingCallback,
530     void                    *Context);


533 /*
534  * dmopcode
535  */
536 void
537 AcpiDmDisassembleOneOp (
538     ACPI_WALK_STATE         *WalkState,
539     ACPI_OP_WALK_INFO       *Info,
540     ACPI_PARSE_OBJECT       *Op);

542 void
543 AcpiDmDecodeInternalObject (
544     ACPI_OPERAND_OBJECT     *ObjDesc);

546 UINT32
547 AcpiDmListType (
548     ACPI_PARSE_OBJECT       *Op);

550 void
551 AcpiDmMethodFlags (
552     ACPI_PARSE_OBJECT       *Op);

554 void
555 AcpiDmPredefinedDescription (
556     ACPI_PARSE_OBJECT       *Op);

558 void
559 AcpiDmFieldPredefinedDescription (
560     ACPI_PARSE_OBJECT       *Op);

562 void
563 AcpiDmFieldFlags (
564     ACPI_PARSE_OBJECT       *Op);

566 void
567 AcpiDmAddressSpace (
568     UINT8                   SpaceId);

570 void
571 AcpiDmRegionFlags (
572     ACPI_PARSE_OBJECT       *Op);

574 void
575 AcpiDmMatchOp (
576     ACPI_PARSE_OBJECT       *Op);


579 /*
580  * dmnames
581  */
582 UINT32
583 AcpiDmDumpName (
584     UINT32                  Name);

586 ACPI_STATUS
587 AcpiPsDisplayObjectPathname (
```



```
588     ACPI_WALK_STATE         *WalkState,
589     ACPI_PARSE_OBJECT       *Op);

591 void
592 AcpiDmNamestring (
593     char                    *Name);


596 /*
597  * dmobject
598  */
599 void
600 AcpiDmDisplayInternalObject (
601     ACPI_OPERAND_OBJECT     *ObjDesc,
602     ACPI_WALK_STATE         *WalkState);

604 void
605 AcpiDmDisplayArguments (
606     ACPI_WALK_STATE         *WalkState);

608 void
609 AcpiDmDisplayLocals (
610     ACPI_WALK_STATE         *WalkState);

612 void
613 AcpiDmDumpMethodInfo (
614     ACPI_STATUS             Status,
615     ACPI_WALK_STATE         *WalkState,
616     ACPI_PARSE_OBJECT       *Op);


619 /*
620  * dmbuffer
621  */
622 void
623 AcpiDmDisasmByteList (
624     UINT32                  Level,
625     UINT8                   *ByteData,
626     UINT32                  ByteCount);

628 void
629 AcpiDmByteList (
630     ACPI_OP_WALK_INFO       *Info,
631     ACPI_PARSE_OBJECT       *Op);

633 void
634 AcpiDmIsEisaId (
635     ACPI_PARSE_OBJECT       *Op);

637 void
638 AcpiDmEisaId (
639     UINT32                  EncodedId);

641 BOOLEAN
642 AcpiDmIsUnicodeBuffer (
643     ACPI_PARSE_OBJECT       *Op);

645 BOOLEAN
646 AcpiDmIsStringBuffer (
647     ACPI_PARSE_OBJECT       *Op);

649 BOOLEAN
650 AcpiDmIsPldBuffer (
651     ACPI_PARSE_OBJECT       *Op);
```

```
654 /*
655  * dmdeferred
656  */
657 ACPI_STATUS
658 AcpiDmParseDeferredOps (
659     ACPI_PARSE_OBJECT       *Root);


662 /*
663  * dmextern
664  */
665 ACPI_STATUS
666 AcpiDmAddToExternalFileList (
667     char                    *PathList);

669 void
670 AcpiDmClearExternalFileList (
671     void);

673 void
674 AcpiDmAddOpToExternalList (
675     ACPI_PARSE_OBJECT       *Op,
676     char                    *Path,
677     UINT8                   Type,
678     UINT32                  Value,
679     UINT16                  Flags);

681 void
682 AcpiDmAddNodeToExternalList (
683     ACPI_NAMESPACE_NODE     *Node,
684     UINT8                   Type,
685     UINT32                  Value,
686     UINT16                  Flags);

688 void
689 AcpiDmAddExternalsToNamespace (
690     void);

692 UINT32
693 AcpiDmGetExternalMethodCount (
694     void);

696 void
697 AcpiDmClearExternalList (
698     void);

700 void
701 AcpiDmEmitExternals (
702     void);

704 void
705 AcpiDmUnresolvedWarning (
706     UINT8                   Type);

708 void
709 AcpiDmGetExternalsFromFile (
710     void);

712 /*
713  * dmresrc
714  */
715 void
716 AcpiDmDumpInteger8 (
717     UINT8                   Value,
718     char                    *Name);
```



```
720 void
721 AcpiDmDumpInteger16 (
722     UINT16                  Value,
723     char                    *Name);

725 void
726 AcpiDmDumpInteger32 (
727     UINT32                  Value,
728     char                    *Name);

730 void
731 AcpiDmDumpInteger64 (
732     UINT64                  Value,
733     char                    *Name);

735 void
736 AcpiDmResourceTemplate (
737     ACPI_OP_WALK_INFO       *Info,
738     ACPI_PARSE_OBJECT       *Op,
739     UINT8                   *ByteData,
740     UINT32                  ByteCount);

742 ACPI_STATUS
743 AcpiDmIsResourceTemplate (
744     ACPI_WALK_STATE         *WalkState,
745     ACPI_PARSE_OBJECT       *Op);

747 void
748 AcpiDmBitList (
749     UINT16                  Mask);

751 void
752 AcpiDmDescriptorName (
753     void);


756 /*
757  * dmresrcl
758  */
759 void
760 AcpiDmWordDescriptor (
761     AML_RESOURCE            *Resource,
762     UINT32                  Length,
763     UINT32                  Level);

765 void
766 AcpiDmDwordDescriptor (
767     AML_RESOURCE            *Resource,
768     UINT32                  Length,
769     UINT32                  Level);

771 void
772 AcpiDmExtendedDescriptor (
773     AML_RESOURCE            *Resource,
774     UINT32                  Length,
775     UINT32                  Level);

777 void
778 AcpiDmQwordDescriptor (
779     AML_RESOURCE            *Resource,
780     UINT32                  Length,
781     UINT32                  Level);

783 void
784 AcpiDmMemory24Descriptor (
785     AML_RESOURCE            *Resource,
```

```
786     UINT32                  Length,
787     UINT32                  Level);

789 void
790 AcpiDmMemory32Descriptor (
791     AML_RESOURCE            *Resource,
792     UINT32                  Length,
793     UINT32                  Level);

795 void
796 AcpiDmFixedMemory32Descriptor (
797     AML_RESOURCE            *Resource,
798     UINT32                  Length,
799     UINT32                  Level);

801 void
802 AcpiDmGenericRegisterDescriptor (
803     AML_RESOURCE            *Resource,
804     UINT32                  Length,
805     UINT32                  Level);

807 void
808 AcpiDmInterruptDescriptor (
809     AML_RESOURCE            *Resource,
810     UINT32                  Length,
811     UINT32                  Level);

813 void
814 AcpiDmVendorLargeDescriptor (
815     AML_RESOURCE            *Resource,
816     UINT32                  Length,
817     UINT32                  Level);

819 void
820 AcpiDmGpioDescriptor (
821     AML_RESOURCE            *Resource,
822     UINT32                  Length,
823     UINT32                  Level);

825 void
826 AcpiDmSerialBusDescriptor (
827     AML_RESOURCE            *Resource,
828     UINT32                  Length,
829     UINT32                  Level);

831 void
832 AcpiDmVendorCommon (
833     char                    *Name,
834     UINT8                   *ByteData,
835     UINT32                  Length,
836     UINT32                  Level);


839 /*
840  * dmresrcs
841  */
842 void
843 AcpiDmIrqDescriptor (
844     AML_RESOURCE            *Resource,
845     UINT32                  Length,
846     UINT32                  Level);

848 void
849 AcpiDmDmaDescriptor (
850     AML_RESOURCE            *Resource,
851     UINT32                  Length,
```



```
852     UINT32                  Level);

854 void
855 AcpiDmFixedDmaDescriptor (
856     AML_RESOURCE            *Resource,
857     UINT32                  Length,
858     UINT32                  Level);

860 void
861 AcpiDmIoDescriptor (
862     AML_RESOURCE            *Resource,
863     UINT32                  Length,
864     UINT32                  Level);

866 void
867 AcpiDmFixedIoDescriptor (
868     AML_RESOURCE            *Resource,
869     UINT32                  Length,
870     UINT32                  Level);

872 void
873 AcpiDmStartDependentDescriptor (
874     AML_RESOURCE            *Resource,
875     UINT32                  Length,
876     UINT32                  Level);

878 void
879 AcpiDmEndDependentDescriptor (
880     AML_RESOURCE            *Resource,
881     UINT32                  Length,
882     UINT32                  Level);

884 void
885 AcpiDmVendorSmallDescriptor (
886     AML_RESOURCE            *Resource,
887     UINT32                  Length,
888     UINT32                  Level);


891 /*
892  * dmutils
893  */
894 void
895 AcpiDmDecodeAttribute (
896     UINT8                   Attribute);

898 void
899 AcpiDmIndent (
900     UINT32                  Level);

902 BOOLEAN
903 AcpiDmCommaIfListMember (
904     ACPI_PARSE_OBJECT       *Op);

906 void
907 AcpiDmCommaIfFieldMember (
908     ACPI_PARSE_OBJECT       *Op);


911 /*
912  * dmrestag
913  */
914 void
915 AcpiDmFindResources (
916     ACPI_PARSE_OBJECT       *Root);
```

```
918 void
919 AcpiDmCheckResourceReference (
920     ACPI_PARSE_OBJECT       *Op,
921     ACPI_WALK_STATE         *WalkState);


924 /*
925  * acdisasm
926  */
927 void
928 AdDisassemblerHeader (
929     char                    *Filename);


932 #endif  /* __ACDISASM_H__ */
```

```
**********************************************************
   11413 Thu Dec 26 13:49:47 2013
new/usr/src/common/acpica/include/acdispat.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acdispat.h - dispatcher (parser to interpreter interface)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef _ACDISPAT_H_
#define _ACDISPAT_H_


#define NAMEOF_LOCAL_NTE    "__L0"
#define NAMEOF_ARG_NTE      "__A0"


/*
 * dsargs - execution of dynamic arguments for static objects
 */
ACPI_STATUS
AcpiDsGetBufferFieldArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc);
```



```
ACPI_STATUS
AcpiDsGetBankFieldArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc);

ACPI_STATUS
AcpiDsGetRegionArguments (
    ACPI_OPERAND_OBJECT     *RgnDesc);

ACPI_STATUS
AcpiDsGetBufferArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc);

ACPI_STATUS
AcpiDsGetPackageArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc);


/*
 * dscontrol - support for execution control opcodes
 */
ACPI_STATUS
AcpiDsExecBeginControlOp (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

ACPI_STATUS
AcpiDsExecEndControlOp (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);


/*
 * dsopcode - support for late operand evaluation
 */
ACPI_STATUS
AcpiDsEvalBufferFieldOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

ACPI_STATUS
AcpiDsEvalRegionOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

ACPI_STATUS
AcpiDsEvalTableRegionOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

ACPI_STATUS
AcpiDsEvalDataObjectOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op,
    ACPI_OPERAND_OBJECT     *ObjDesc);

ACPI_STATUS
AcpiDsEvalBankFieldOperands (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op);

ACPI_STATUS
AcpiDsInitializeRegion (
    ACPI_HANDLE             ObjHandle);


/*
```

```
126  * dsexec - Parser/Interpreter interface, method execution callbacks
127  */
128 ACPI_STATUS
129 AcpiDsGetPredicateValue (
130     ACPI_WALK_STATE         *WalkState,
131     ACPI_OPERAND_OBJECT     *ResultObj);

133 ACPI_STATUS
134 AcpiDsExecBeginOp (
135     ACPI_WALK_STATE         *WalkState,
136     ACPI_PARSE_OBJECT       **OutOp);

138 ACPI_STATUS
139 AcpiDsExecEndOp (
140     ACPI_WALK_STATE         *State);


143 /*
144  * dsfield - Parser/Interpreter interface for AML fields
145  */
146 ACPI_STATUS
147 AcpiDsCreateField (
148     ACPI_PARSE_OBJECT       *Op,
149     ACPI_NAMESPACE_NODE     *RegionNode,
150     ACPI_WALK_STATE         *WalkState);

152 ACPI_STATUS
153 AcpiDsCreateBankField (
154     ACPI_PARSE_OBJECT       *Op,
155     ACPI_NAMESPACE_NODE     *RegionNode,
156     ACPI_WALK_STATE         *WalkState);

158 ACPI_STATUS
159 AcpiDsCreateIndexField (
160     ACPI_PARSE_OBJECT       *Op,
161     ACPI_NAMESPACE_NODE     *RegionNode,
162     ACPI_WALK_STATE         *WalkState);

164 ACPI_STATUS
165 AcpiDsCreateBufferField (
166     ACPI_PARSE_OBJECT       *Op,
167     ACPI_WALK_STATE         *WalkState);

169 ACPI_STATUS
170 AcpiDsInitFieldObjects (
171     ACPI_PARSE_OBJECT       *Op,
172     ACPI_WALK_STATE         *WalkState);


175 /*
176  * dsload - Parser/Interpreter interface, pass 1 namespace load callbacks
177  */
178 ACPI_STATUS
179 AcpiDsInitCallbacks (
180     ACPI_WALK_STATE         *WalkState,
181     UINT32                  PassNumber);

183 ACPI_STATUS
184 AcpiDsLoad1BeginOp (
185     ACPI_WALK_STATE         *WalkState,
186     ACPI_PARSE_OBJECT       **OutOp);

188 ACPI_STATUS
189 AcpiDsLoad1EndOp (
190     ACPI_WALK_STATE         *WalkState);
```



```
193 /*
194  * dsload - Parser/Interpreter interface, pass 2 namespace load callbacks
195  */
196 ACPI_STATUS
197 AcpiDsLoad2BeginOp (
198     ACPI_WALK_STATE         *WalkState,
199     ACPI_PARSE_OBJECT       **OutOp);

201 ACPI_STATUS
202 AcpiDsLoad2EndOp (
203     ACPI_WALK_STATE         *WalkState);


206 /*
207  * dsmthdat - method data (locals/args)
208  */
209 ACPI_STATUS
210 AcpiDsStoreObjectToLocal (
211     UINT8                   Type,
212     UINT32                  Index,
213     ACPI_OPERAND_OBJECT     *SrcDesc,
214     ACPI_WALK_STATE         *WalkState);

216 ACPI_STATUS
217 AcpiDsMethodDataGetEntry (
218     UINT16                  Opcode,
219     UINT32                  Index,
220     ACPI_WALK_STATE         *WalkState,
221     ACPI_OPERAND_OBJECT     ***Node);

223 void
224 AcpiDsMethodDataDeleteAll (
225     ACPI_WALK_STATE         *WalkState);

227 BOOLEAN
228 AcpiDsIsMethodValue (
229     ACPI_OPERAND_OBJECT     *ObjDesc);

231 ACPI_STATUS
232 AcpiDsMethodDataGetValue (
233     UINT8                   Type,
234     UINT32                  Index,
235     ACPI_WALK_STATE         *WalkState,
236     ACPI_OPERAND_OBJECT     **DestDesc);

238 ACPI_STATUS
239 AcpiDsMethodDataInitArgs (
240     ACPI_OPERAND_OBJECT     **Params,
241     UINT32                  MaxParamCount,
242     ACPI_WALK_STATE         *WalkState);

244 ACPI_STATUS
245 AcpiDsMethodDataGetNode (
246     UINT8                   Type,
247     UINT32                  Index,
248     ACPI_WALK_STATE         *WalkState,
249     ACPI_NAMESPACE_NODE     **Node);

251 void
252 AcpiDsMethodDataInit (
253     ACPI_WALK_STATE         *WalkState);


256 /*
257  * dsmethod - Parser/Interpreter interface - control method parsing
```

```
 258  */
 259 ACPI_STATUS
 260 AcpiDsParseMethod (
 261     ACPI_NAMESPACE_NODE     *Node);

 263 ACPI_STATUS
 264 AcpiDsCallControlMethod (
 265     ACPI_THREAD_STATE       *Thread,
 266     ACPI_WALK_STATE         *WalkState,
 267     ACPI_PARSE_OBJECT       *Op);

 269 ACPI_STATUS
 270 AcpiDsRestartControlMethod (
 271     ACPI_WALK_STATE         *WalkState,
 272     ACPI_OPERAND_OBJECT     *ReturnDesc);

 274 void
 275 AcpiDsTerminateControlMethod (
 276     ACPI_OPERAND_OBJECT     *MethodDesc,
 277     ACPI_WALK_STATE         *WalkState);

 279 ACPI_STATUS
 280 AcpiDsBeginMethodExecution (
 281     ACPI_NAMESPACE_NODE     *MethodNode,
 282     ACPI_OPERAND_OBJECT     *ObjDesc,
 283     ACPI_WALK_STATE         *WalkState);

 285 ACPI_STATUS
 286 AcpiDsMethodError (
 287     ACPI_STATUS             Status,
 288     ACPI_WALK_STATE         *WalkState);

 290 /*
 291  * dsinit
 292  */
 293 ACPI_STATUS
 294 AcpiDsInitializeObjects (
 295     UINT32                  TableIndex,
 296     ACPI_NAMESPACE_NODE     *StartNode);


 299 /*
 300  * dsobject - Parser/Interpreter interface - object initialization and conversio
 301  */
 302 ACPI_STATUS
 303 AcpiDsBuildInternalBufferObj (
 304     ACPI_WALK_STATE         *WalkState,
 305     ACPI_PARSE_OBJECT       *Op,
 306     UINT32                  BufferLength,
 307     ACPI_OPERAND_OBJECT     **ObjDescPtr);

 309 ACPI_STATUS
 310 AcpiDsBuildInternalPackageObj (
 311     ACPI_WALK_STATE         *WalkState,
 312     ACPI_PARSE_OBJECT       *op,
 313     UINT32                  PackageLength,
 314     ACPI_OPERAND_OBJECT     **ObjDesc);

 316 ACPI_STATUS
 317 AcpiDsInitObjectFromOp (
 318     ACPI_WALK_STATE         *WalkState,
 319     ACPI_PARSE_OBJECT       *Op,
 320     UINT16                  Opcode,
 321     ACPI_OPERAND_OBJECT     **ObjDesc);

 323 ACPI_STATUS
```



```
 324 AcpiDsCreateNode (
 325     ACPI_WALK_STATE         *WalkState,
 326     ACPI_NAMESPACE_NODE     *Node,
 327     ACPI_PARSE_OBJECT       *Op);


 330 /*
 331  * dsutils - Parser/Interpreter interface utility routines
 332  */
 333 void
 334 AcpiDsClearImplicitReturn (
 335     ACPI_WALK_STATE         *WalkState);

 337 BOOLEAN
 338 AcpiDsDoImplicitReturn (
 339     ACPI_OPERAND_OBJECT     *ReturnDesc,
 340     ACPI_WALK_STATE         *WalkState,
 341     BOOLEAN                 AddReference);

 343 BOOLEAN
 344 AcpiDsIsResultUsed (
 345     ACPI_PARSE_OBJECT       *Op,
 346     ACPI_WALK_STATE         *WalkState);

 348 void
 349 AcpiDsDeleteResultIfNotUsed (
 350     ACPI_PARSE_OBJECT       *Op,
 351     ACPI_OPERAND_OBJECT     *ResultObj,
 352     ACPI_WALK_STATE         *WalkState);

 354 ACPI_STATUS
 355 AcpiDsCreateOperand (
 356     ACPI_WALK_STATE         *WalkState,
 357     ACPI_PARSE_OBJECT       *Arg,
 358     UINT32                  ArgsRemaining);

 360 ACPI_STATUS
 361 AcpiDsCreateOperands (
 362     ACPI_WALK_STATE         *WalkState,
 363     ACPI_PARSE_OBJECT       *FirstArg);

 365 ACPI_STATUS
 366 AcpiDsResolveOperands (
 367     ACPI_WALK_STATE         *WalkState);

 369 void
 370 AcpiDsClearOperands (
 371     ACPI_WALK_STATE         *WalkState);

 373 ACPI_STATUS
 374 AcpiDsEvaluateNamePath (
 375     ACPI_WALK_STATE         *WalkState);


 378 /*
 379  * dswscope - Scope Stack manipulation
 380  */
 381 ACPI_STATUS
 382 AcpiDsScopeStackPush (
 383     ACPI_NAMESPACE_NODE     *Node,
 384     ACPI_OBJECT_TYPE        Type,
 385     ACPI_WALK_STATE         *WalkState);


 388 ACPI_STATUS
 389 AcpiDsScopeStackPop (
```

```
390     ACPI_WALK_STATE         *WalkState);

392 void
393 AcpiDsScopeStackClear (
394     ACPI_WALK_STATE         *WalkState);


397 /*
398  * dswstate - parser WALK_STATE management routines
399  */
400 ACPI_STATUS
401 AcpiDsObjStackPush (
402     void                    *Object,
403     ACPI_WALK_STATE         *WalkState);

405 ACPI_STATUS
406 AcpiDsObjStackPop (
407     UINT32                  PopCount,
408     ACPI_WALK_STATE         *WalkState);

410 ACPI_WALK_STATE *
411 AcpiDsCreateWalkState (
412     ACPI_OWNER_ID           OwnerId,
413     ACPI_PARSE_OBJECT       *Origin,
414     ACPI_OPERAND_OBJECT     *MthDesc,
415     ACPI_THREAD_STATE       *Thread);

417 ACPI_STATUS
418 AcpiDsInitAmlWalk (
419     ACPI_WALK_STATE         *WalkState,
420     ACPI_PARSE_OBJECT       *Op,
421     ACPI_NAMESPACE_NODE     *MethodNode,
422     UINT8                   *AmlStart,
423     UINT32                  AmlLength,
424     ACPI_EVALUATE_INFO      *Info,
425     UINT8                   PassNumber);

427 void
428 AcpiDsObjStackPopAndDelete (
429     UINT32                  PopCount,
430     ACPI_WALK_STATE         *WalkState);

432 void
433 AcpiDsDeleteWalkState (
434     ACPI_WALK_STATE         *WalkState);

436 ACPI_WALK_STATE *
437 AcpiDsPopWalkState (
438     ACPI_THREAD_STATE       *Thread);

440 void
441 AcpiDsPushWalkState (
442     ACPI_WALK_STATE         *WalkState,
443     ACPI_THREAD_STATE       *Thread);

445 ACPI_STATUS
446 AcpiDsResultStackClear (
447     ACPI_WALK_STATE         *WalkState);

449 ACPI_WALK_STATE *
450 AcpiDsGetCurrentWalkState (
451     ACPI_THREAD_STATE       *Thread);

453 ACPI_STATUS
454 AcpiDsResultPop (
455     ACPI_OPERAND_OBJECT     **Object,
```



```
456     ACPI_WALK_STATE         *WalkState);

458 ACPI_STATUS
459 AcpiDsResultPush (
460     ACPI_OPERAND_OBJECT     *Object,
461     ACPI_WALK_STATE         *WalkState);

463 #endif /* _ACDISPAT_H_ */
```

```
**********************************************************
    8717 Thu Dec 26 13:49:48 2013
new/usr/src/common/acpica/include/acevents.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acevents.h - Event subcomponent prototypes and defines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACEVENTS_H__
#define __ACEVENTS_H__


/*
 * evevent
 */
ACPI_STATUS
AcpiEvInitializeEvents (
    void);

ACPI_STATUS
AcpiEvInstallXruptHandlers (
    void);
```



```
UINT32
AcpiEvFixedEventDetect (
    void);


/*
 * evmisc
 */
BOOLEAN
AcpiEvIsNotifyObject (
    ACPI_NAMESPACE_NODE     *Node);

UINT32
AcpiEvGetGpeNumberIndex (
    UINT32                  GpeNumber);

ACPI_STATUS
AcpiEvQueueNotifyRequest (
    ACPI_NAMESPACE_NODE     *Node,
    UINT32                  NotifyValue);


/*
 * evglock - Global Lock support
 */
ACPI_STATUS
AcpiEvInitGlobalLockHandler (
    void);

ACPI_HW_DEPENDENT_RETURN_OK (
ACPI_STATUS
AcpiEvAcquireGlobalLock(
    UINT16                  Timeout))
    UINT16                  Timeout);

ACPI_HW_DEPENDENT_RETURN_OK (
ACPI_STATUS
AcpiEvReleaseGlobalLock(
    void))
    void);

ACPI_STATUS
AcpiEvRemoveGlobalLockHandler (
    void);


/*
 * evgpe - Low-level GPE support
 */
UINT32
AcpiEvGpeDetect (
    ACPI_GPE_XRUPT_INFO     *GpeXruptList);

ACPI_STATUS
AcpiEvUpdateGpeEnableMask (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);

ACPI_STATUS
AcpiEvEnableGpe (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);

ACPI_STATUS
AcpiEvAddGpeReference (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);

ACPI_STATUS
```

```
AcpiEvRemoveGpeReference (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);

ACPI_GPE_EVENT_INFO *
AcpiEvGetGpeEventInfo (
    ACPI_HANDLE             GpeDevice,
    UINT32                  GpeNumber);

ACPI_GPE_EVENT_INFO *
AcpiEvLowGetGpeInfo (
    UINT32                  GpeNumber,
    ACPI_GPE_BLOCK_INFO     *GpeBlock);

ACPI_STATUS
AcpiEvFinishGpe (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);


/*
 * evgpeblk - Upper-level GPE block support
 */
ACPI_STATUS
AcpiEvCreateGpeBlock (
    ACPI_NAMESPACE_NODE     *GpeDevice,
    ACPI_GENERIC_ADDRESS    *GpeBlockAddress,
    UINT32                  RegisterCount,
    UINT8                   GpeBlockBaseNumber,
    UINT32                  InterruptNumber,
    ACPI_GPE_BLOCK_INFO     **ReturnGpeBlock);

ACPI_STATUS
AcpiEvInitializeGpeBlock (
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    void                    *Context);

ACPI_HW_DEPENDENT_RETURN_OK (
ACPI_STATUS
AcpiEvDeleteGpeBlock (
    ACPI_GPE_BLOCK_INFO     *GpeBlock))
    ACPI_GPE_BLOCK_INFO     *GpeBlock);

UINT32
AcpiEvGpeDispatch (
    ACPI_NAMESPACE_NODE     *GpeDevice,
    ACPI_GPE_EVENT_INFO     *GpeEventInfo,
    UINT32                  GpeNumber);


/*
 * evgpeinit - GPE initialization and update
 */
ACPI_STATUS
AcpiEvGpeInitialize (
    void);

ACPI_HW_DEPENDENT_RETURN_VOID (
void
AcpiEvUpdateGpes (
    ACPI_OWNER_ID           TableOwnerId))
    ACPI_OWNER_ID           TableOwnerId);

ACPI_STATUS
AcpiEvMatchGpeMethod (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
```



```
    void                    *Context,
    void                    **ReturnValue);


/*
 * evgpeutil - GPE utilities
 */
ACPI_STATUS
AcpiEvWalkGpeList (
    ACPI_GPE_CALLBACK       GpeWalkCallback,
    void                    *Context);

BOOLEAN
AcpiEvValidGpeEvent (
    ACPI_GPE_EVENT_INFO     *GpeEventInfo);

ACPI_STATUS
AcpiEvGetGpeDevice (
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    void                    *Context);

ACPI_STATUS
ACPI_GPE_XRUPT_INFO *
AcpiEvGetGpeXruptBlock (
    UINT32                  InterruptNumber,
    ACPI_GPE_XRUPT_INFO     **GpeXruptBlock);
    UINT32                  InterruptNumber);

ACPI_STATUS
AcpiEvDeleteGpeXrupt (
    ACPI_GPE_XRUPT_INFO     *GpeXrupt);

ACPI_STATUS
AcpiEvDeleteGpeHandlers (
    ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
    ACPI_GPE_BLOCK_INFO     *GpeBlock,
    void                    *Context);


/*
 * evhandler - Address space handling
 * evregion - Address Space handling
 */
BOOLEAN
AcpiEvHasDefaultHandler (
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_ADR_SPACE_TYPE     SpaceId);

ACPI_STATUS
AcpiEvInstallRegionHandlers (
    void);

ACPI_STATUS
AcpiEvInstallSpaceHandler (
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_ADR_SPACE_TYPE     SpaceId,
    ACPI_ADR_SPACE_HANDLER  Handler,
    ACPI_ADR_SPACE_SETUP    Setup,
    void                    *Context);


/*
 * evregion - Operation region support
 */
ACPI_STATUS
```

```
250 AcpiEvInitializeOpRegions (
251     void);

253 ACPI_STATUS
254 AcpiEvAddressSpaceDispatch (
255     ACPI_OPERAND_OBJECT     *RegionObj,
256     ACPI_OPERAND_OBJECT     *FieldObj,
257     UINT32                  Function,
258     UINT32                  RegionOffset,
259     UINT32                  BitWidth,
260     UINT64                  *Value);

262 ACPI_STATUS
263 AcpiEvAttachRegion (
264     ACPI_OPERAND_OBJECT     *HandlerObj,
265     ACPI_OPERAND_OBJECT     *RegionObj,
266     BOOLEAN                 AcpiNsIsLocked);

268 void
269 AcpiEvDetachRegion (
270     ACPI_OPERAND_OBJECT    *RegionObj,
271     BOOLEAN                 AcpiNsIsLocked);

273 ACPI_STATUS
249 AcpiEvInstallSpaceHandler (
250     ACPI_NAMESPACE_NODE     *Node,
251     ACPI_ADR_SPACE_TYPE     SpaceId,
252     ACPI_ADR_SPACE_HANDLER  Handler,
253     ACPI_ADR_SPACE_SETUP    Setup,
254     void                    *Context);

256 ACPI_STATUS
274 AcpiEvExecuteRegMethods (
275     ACPI_NAMESPACE_NODE     *Node,
276     ACPI_ADR_SPACE_TYPE     SpaceId);

278 ACPI_STATUS
279 AcpiEvExecuteRegMethod (
280     ACPI_OPERAND_OBJECT    *RegionObj,
281     UINT32                  Function);


284 /*
285  * evregini - Region initialization and setup
286  */
287 ACPI_STATUS
288 AcpiEvSystemMemoryRegionSetup (
289     ACPI_HANDLE             Handle,
290     UINT32                  Function,
291     void                    *HandlerContext,
292     void                    **RegionContext);

294 ACPI_STATUS
295 AcpiEvIoSpaceRegionSetup (
296     ACPI_HANDLE             Handle,
297     UINT32                  Function,
298     void                    *HandlerContext,
299     void                    **RegionContext);

301 ACPI_STATUS
302 AcpiEvPciConfigRegionSetup (
303     ACPI_HANDLE             Handle,
304     UINT32                  Function,
305     void                    *HandlerContext,
306     void                    **RegionContext);
```



```
308 ACPI_STATUS
309 AcpiEvCmosRegionSetup (
310     ACPI_HANDLE             Handle,
311     UINT32                  Function,
312     void                    *HandlerContext,
313     void                    **RegionContext);

315 ACPI_STATUS
316 AcpiEvPciBarRegionSetup (
317     ACPI_HANDLE             Handle,
318     UINT32                  Function,
319     void                    *HandlerContext,
320     void                    **RegionContext);

322 ACPI_STATUS
323 AcpiEvDefaultRegionSetup (
324     ACPI_HANDLE             Handle,
325     UINT32                  Function,
326     void                    *HandlerContext,
327     void                    **RegionContext);

329 ACPI_STATUS
330 AcpiEvInitializeRegion (
331     ACPI_OPERAND_OBJECT     *RegionObj,
332     BOOLEAN                 AcpiNsLocked);


335 /*
336  * evsci - SCI (System Control Interrupt) handling/dispatch
337  */
338 UINT32 ACPI_SYSTEM_XFACE
339 AcpiEvGpeXruptHandler (
340     void                    *Context);

342 UINT32
343 AcpiEvSciDispatch (
344     void);

346 UINT32
347 AcpiEvInstallSciHandler (
348     void);

350 ACPI_STATUS
351 AcpiEvRemoveAllSciHandlers (
330 AcpiEvRemoveSciHandler (
352     void);

354 ACPI_HW_DEPENDENT_RETURN_VOID (
333 UINT32
334 AcpiEvInitializeSCI (
335     UINT32                  ProgramSCI);

355 void
356 AcpiEvTerminate (
357     void))
339     void);


359 #endif  /* __ACEVENTS_H__  */
```

```
**********************************************************
   18604 Thu Dec 26 13:49:48 2013
new/usr/src/common/acpica/include/acexcep.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acexcep.h - Exception codes returned by the ACPI subsystem
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACEXCEP_H__
#define __ACEXCEP_H__


/* This module contains all possible exception codes for ACPI_STATUS */

/*
 * Exception code classes
 */
#define AE_CODE_ENVIRONMENTAL           0x0000 /* General ACPICA environment */
#define AE_CODE_PROGRAMMER              0x1000 /* External ACPICA interface call
#define AE_CODE_ACPI_TABLES             0x2000 /* ACPI tables */
#define AE_CODE_AML                     0x3000 /* From executing AML code */
#define AE_CODE_CONTROL                 0x4000 /* Internal control codes */

#define AE_CODE_MAX                     0x4000
#define AE_CODE_MASK                    0xF000
```



```
/*
 * Macros to insert the exception code classes
 */
#define EXCEP_ENV(code)                 ((ACPI_STATUS) (code | AE_CODE_ENVIRONME
#define EXCEP_PGM(code)                 ((ACPI_STATUS) (code | AE_CODE_PROGRAMME
#define EXCEP_TBL(code)                 ((ACPI_STATUS) (code | AE_CODE_ACPI_TABL
#define EXCEP_AML(code)                 ((ACPI_STATUS) (code | AE_CODE_AML))
#define EXCEP_CTL(code)                 ((ACPI_STATUS) (code | AE_CODE_CONTROL))

/*
 * Exception info table. The "Description" field is used only by the
 * ACPICA help application (acpihelp).
 */
typedef struct acpi_exception_info
{
    char                *Name;

#ifdef ACPI_HELP_APP
    char                *Description;
#endif
} ACPI_EXCEPTION_INFO;

#ifdef ACPI_HELP_APP
#define EXCEP_TXT(Name,Description)     {Name, Description}
#else
#define EXCEP_TXT(Name,Description)     {Name}
#endif


/*
 * Success is always zero, failure is non-zero
 */
#define ACPI_SUCCESS(a)                 (!(a))
#define ACPI_FAILURE(a)                 (a)

#define AE_OK                           (ACPI_STATUS) 0x0000

/*
 * Environmental exceptions
 */
#define AE_ERROR                        EXCEP_ENV (0x0001)
#define AE_NO_ACPI_TABLES               EXCEP_ENV (0x0002)
#define AE_NO_NAMESPACE                 EXCEP_ENV (0x0003)
#define AE_NO_MEMORY                    EXCEP_ENV (0x0004)
#define AE_NOT_FOUND                    EXCEP_ENV (0x0005)
#define AE_NOT_EXIST                    EXCEP_ENV (0x0006)
#define AE_ALREADY_EXISTS               EXCEP_ENV (0x0007)
#define AE_TYPE                         EXCEP_ENV (0x0008)
#define AE_NULL_OBJECT                  EXCEP_ENV (0x0009)
#define AE_NULL_ENTRY                   EXCEP_ENV (0x000A)
#define AE_BUFFER_OVERFLOW              EXCEP_ENV (0x000B)
#define AE_STACK_OVERFLOW               EXCEP_ENV (0x000C)
#define AE_STACK_UNDERFLOW              EXCEP_ENV (0x000D)
#define AE_NOT_IMPLEMENTED              EXCEP_ENV (0x000E)
#define AE_SUPPORT                      EXCEP_ENV (0x000F)
#define AE_LIMIT                        EXCEP_ENV (0x0010)
#define AE_TIME                         EXCEP_ENV (0x0011)
#define AE_ACQUIRE_DEADLOCK             EXCEP_ENV (0x0012)
#define AE_RELEASE_DEADLOCK             EXCEP_ENV (0x0013)
#define AE_NOT_ACQUIRED                 EXCEP_ENV (0x0014)
#define AE_ALREADY_ACQUIRED             EXCEP_ENV (0x0015)
#define AE_NO_HARDWARE_RESPONSE         EXCEP_ENV (0x0016)
#define AE_NO_GLOBAL_LOCK               EXCEP_ENV (0x0017)
#define AE_ABORT_METHOD                 EXCEP_ENV (0x0018)
#define AE_SAME_HANDLER                 EXCEP_ENV (0x0019)
```

```
127 #define AE_NO_HANDLER                   EXCEP_ENV (0x001A)
128 #define AE_OWNER_ID_LIMIT               EXCEP_ENV (0x001B)
129 #define AE_NOT_CONFIGURED               EXCEP_ENV (0x001C)
130 #define AE_ACCESS                       EXCEP_ENV (0x001D)

132 #define AE_CODE_ENV_MAX                 0x001D


135 /*
136  * Programmer exceptions
137  */
138 #define AE_BAD_PARAMETER                EXCEP_PGM (0x0001)
139 #define AE_BAD_CHARACTER                EXCEP_PGM (0x0002)
140 #define AE_BAD_PATHNAME                 EXCEP_PGM (0x0003)
141 #define AE_BAD_DATA                     EXCEP_PGM (0x0004)
142 #define AE_BAD_HEX_CONSTANT             EXCEP_PGM (0x0005)
143 #define AE_BAD_OCTAL_CONSTANT           EXCEP_PGM (0x0006)
144 #define AE_BAD_DECIMAL_CONSTANT         EXCEP_PGM (0x0007)
145 #define AE_MISSING_ARGUMENTS            EXCEP_PGM (0x0008)
146 #define AE_BAD_ADDRESS                  EXCEP_PGM (0x0009)

148 #define AE_CODE_PGM_MAX                 0x0009


151 /*
152  * Acpi table exceptions
153  */
154 #define AE_BAD_SIGNATURE                EXCEP_TBL (0x0001)
155 #define AE_BAD_HEADER                   EXCEP_TBL (0x0002)
156 #define AE_BAD_CHECKSUM                 EXCEP_TBL (0x0003)
157 #define AE_BAD_VALUE                    EXCEP_TBL (0x0004)
158 #define AE_INVALID_TABLE_LENGTH         EXCEP_TBL (0x0005)

160 #define AE_CODE_TBL_MAX                 0x0005


163 /*
164  * AML exceptions. These are caused by problems with
165  * the actual AML byte stream
166  */
167 #define AE_AML_BAD_OPCODE               EXCEP_AML (0x0001)
168 #define AE_AML_NO_OPERAND               EXCEP_AML (0x0002)
169 #define AE_AML_OPERAND_TYPE             EXCEP_AML (0x0003)
170 #define AE_AML_OPERAND_VALUE            EXCEP_AML (0x0004)
171 #define AE_AML_UNINITIALIZED_LOCAL      EXCEP_AML (0x0005)
172 #define AE_AML_UNINITIALIZED_ARG        EXCEP_AML (0x0006)
173 #define AE_AML_UNINITIALIZED_ELEMENT    EXCEP_AML (0x0007)
174 #define AE_AML_NUMERIC_OVERFLOW         EXCEP_AML (0x0008)
175 #define AE_AML_REGION_LIMIT             EXCEP_AML (0x0009)
176 #define AE_AML_BUFFER_LIMIT             EXCEP_AML (0x000A)
177 #define AE_AML_PACKAGE_LIMIT            EXCEP_AML (0x000B)
178 #define AE_AML_DIVIDE_BY_ZERO           EXCEP_AML (0x000C)
179 #define AE_AML_BAD_NAME                 EXCEP_AML (0x000D)
180 #define AE_AML_NAME_NOT_FOUND           EXCEP_AML (0x000E)
181 #define AE_AML_INTERNAL                 EXCEP_AML (0x000F)
182 #define AE_AML_INVALID_SPACE_ID         EXCEP_AML (0x0010)
183 #define AE_AML_STRING_LIMIT             EXCEP_AML (0x0011)
184 #define AE_AML_NO_RETURN_VALUE          EXCEP_AML (0x0012)
185 #define AE_AML_METHOD_LIMIT             EXCEP_AML (0x0013)
186 #define AE_AML_NOT_OWNER                EXCEP_AML (0x0014)
187 #define AE_AML_MUTEX_ORDER              EXCEP_AML (0x0015)
188 #define AE_AML_MUTEX_NOT_ACQUIRED       EXCEP_AML (0x0016)
189 #define AE_AML_INVALID_RESOURCE_TYPE    EXCEP_AML (0x0017)
190 #define AE_AML_INVALID_INDEX            EXCEP_AML (0x0018)
191 #define AE_AML_REGISTER_LIMIT           EXCEP_AML (0x0019)
192 #define AE_AML_NO_WHILE                 EXCEP_AML (0x001A)
```



```
193 #define AE_AML_ALIGNMENT                EXCEP_AML (0x001B)
194 #define AE_AML_NO_RESOURCE_END_TAG      EXCEP_AML (0x001C)
195 #define AE_AML_BAD_RESOURCE_VALUE       EXCEP_AML (0x001D)
196 #define AE_AML_CIRCULAR_REFERENCE       EXCEP_AML (0x001E)
197 #define AE_AML_BAD_RESOURCE_LENGTH      EXCEP_AML (0x001F)
198 #define AE_AML_ILLEGAL_ADDRESS          EXCEP_AML (0x0020)
199 #define AE_AML_INFINITE_LOOP            EXCEP_AML (0x0021)

201 #define AE_CODE_AML_MAX                 0x0021


204 /*
205  * Internal exceptions used for control
206  */
207 #define AE_CTRL_RETURN_VALUE            EXCEP_CTL (0x0001)
208 #define AE_CTRL_PENDING                 EXCEP_CTL (0x0002)
209 #define AE_CTRL_TERMINATE               EXCEP_CTL (0x0003)
210 #define AE_CTRL_TRUE                    EXCEP_CTL (0x0004)
211 #define AE_CTRL_FALSE                   EXCEP_CTL (0x0005)
212 #define AE_CTRL_DEPTH                   EXCEP_CTL (0x0006)
213 #define AE_CTRL_END                     EXCEP_CTL (0x0007)
214 #define AE_CTRL_TRANSFER                EXCEP_CTL (0x0008)
215 #define AE_CTRL_BREAK                   EXCEP_CTL (0x0009)
216 #define AE_CTRL_CONTINUE                EXCEP_CTL (0x000A)
217 #define AE_CTRL_SKIP                    EXCEP_CTL (0x000B)
218 #define AE_CTRL_PARSE_CONTINUE          EXCEP_CTL (0x000C)
219 #define AE_CTRL_PARSE_PENDING           EXCEP_CTL (0x000D)

221 #define AE_CODE_CTRL_MAX                0x000D


224 /* Exception strings for AcpiFormatException */

226 #ifdef ACPI_DEFINE_EXCEPTION_TABLE

228 /*
229  * String versions of the exception codes above
230  * These strings must match the corresponding defines exactly
231  */
232 static const ACPI_EXCEPTION_INFO    AcpiGbl_ExceptionNames_Env[] =
233 {
234     EXCEP_TXT ("AE_OK",                         "No error"),
235     EXCEP_TXT ("AE_ERROR",                      "Unspecified error"),
236     EXCEP_TXT ("AE_NO_ACPI_TABLES",             "ACPI tables could not be found"
237     EXCEP_TXT ("AE_NO_NAMESPACE",               "A namespace has not been loaded
238     EXCEP_TXT ("AE_NO_MEMORY",                  "Insufficient dynamic memory"),
239     EXCEP_TXT ("AE_NOT_FOUND",                  "A requested entity is not found
240     EXCEP_TXT ("AE_NOT_EXIST",                  "A required entity does not exis
241     EXCEP_TXT ("AE_ALREADY_EXISTS",             "An entity already exists"),
242     EXCEP_TXT ("AE_TYPE",                       "The object type is incorrect"),
243     EXCEP_TXT ("AE_NULL_OBJECT",                "A required object was missing")
244     EXCEP_TXT ("AE_NULL_ENTRY",                 "The requested object does not e
245     EXCEP_TXT ("AE_BUFFER_OVERFLOW",            "The buffer provided is too smal
246     EXCEP_TXT ("AE_STACK_OVERFLOW",             "An internal stack overflowed"),
247     EXCEP_TXT ("AE_STACK_UNDERFLOW",            "An internal stack underflowed")
248     EXCEP_TXT ("AE_NOT_IMPLEMENTED",            "The feature is not implemented"
249     EXCEP_TXT ("AE_SUPPORT",                    "The feature is not supported"),
250     EXCEP_TXT ("AE_LIMIT",                      "A predefined limit was exceeded
251     EXCEP_TXT ("AE_TIME",                       "A time limit or timeout expired
252     EXCEP_TXT ("AE_ACQUIRE_DEADLOCK",           "Internal error, attempt was mad
253     EXCEP_TXT ("AE_RELEASE_DEADLOCK",           "Internal error, attempt was mad
254     EXCEP_TXT ("AE_NOT_ACQUIRED",               "An attempt to release a mutex o
255     EXCEP_TXT ("AE_ALREADY_ACQUIRED",           "Internal error, attempt was mad
256     EXCEP_TXT ("AE_NO_HARDWARE_RESPONSE",       "Hardware did not respond after
257     EXCEP_TXT ("AE_NO_GLOBAL_LOCK",             "There is no FACS Global Lock"),
258     EXCEP_TXT ("AE_ABORT_METHOD",               "A control method was aborted"),
```

```
259     EXCEP_TXT ("AE_SAME_HANDLER",               "Attempt was made to install the
260     EXCEP_TXT ("AE_NO_HANDLER",                 "A handler for the operation is
261     EXCEP_TXT ("AE_OWNER_ID_LIMIT",             "There are no more Owner IDs ava
262     EXCEP_TXT ("AE_NOT_CONFIGURED",             "The interface is not part of th
263     EXCEP_TXT ("AE_ACCESS",                     "Permission denied for the reque
264 };

266 static const ACPI_EXCEPTION_INFO    AcpiGbl_ExceptionNames_Pgm[] =
267 {
268     EXCEP_TXT (NULL, NULL),
269     EXCEP_TXT ("AE_BAD_PARAMETER",              "A parameter is out of range or
270     EXCEP_TXT ("AE_BAD_CHARACTER",              "An invalid character was found
271     EXCEP_TXT ("AE_BAD_PATHNAME",               "An invalid character was found
272     EXCEP_TXT ("AE_BAD_DATA",                   "A package or buffer contained i
273     EXCEP_TXT ("AE_BAD_HEX_CONSTANT",           "Invalid character in a Hex cons
274     EXCEP_TXT ("AE_BAD_OCTAL_CONSTANT",         "Invalid character in an Octal c
275     EXCEP_TXT ("AE_BAD_DECIMAL_CONSTANT",       "Invalid character in a Decimal
276     EXCEP_TXT ("AE_MISSING_ARGUMENTS",          "Too few arguments were passed t
277     EXCEP_TXT ("AE_BAD_ADDRESS",                "An illegal null I/O address")
278 };

280 static const ACPI_EXCEPTION_INFO    AcpiGbl_ExceptionNames_Tbl[] =
281 {
282     EXCEP_TXT (NULL, NULL),
283     EXCEP_TXT ("AE_BAD_SIGNATURE",              "An ACPI table has an invalid si
284     EXCEP_TXT ("AE_BAD_HEADER",                 "Invalid field in an ACPI table
285     EXCEP_TXT ("AE_BAD_CHECKSUM",               "An ACPI table checksum is not c
286     EXCEP_TXT ("AE_BAD_VALUE",                  "An invalid value was found in a
287     EXCEP_TXT ("AE_INVALID_TABLE_LENGTH",       "The FADT or FACS has improper l
288 };

290 static const ACPI_EXCEPTION_INFO    AcpiGbl_ExceptionNames_Aml[] =
291 {
292     EXCEP_TXT (NULL, NULL),
293     EXCEP_TXT ("AE_AML_BAD_OPCODE",             "Invalid AML opcode encountered"
294     EXCEP_TXT ("AE_AML_NO_OPERAND",             "A required operand is missing")
295     EXCEP_TXT ("AE_AML_OPERAND_TYPE",           "An operand of an incorrect type
296     EXCEP_TXT ("AE_AML_OPERAND_VALUE",          "The operand had an inappropriat
297     EXCEP_TXT ("AE_AML_UNINITIALIZED_LOCAL",    "Method tried to use an uninitia
298     EXCEP_TXT ("AE_AML_UNINITIALIZED_ARG",      "Method tried to use an uninitia
299     EXCEP_TXT ("AE_AML_UNINITIALIZED_ELEMENT",  "Method tried to use an empty pa
300     EXCEP_TXT ("AE_AML_NUMERIC_OVERFLOW",       "Overflow during BCD conversion
301     EXCEP_TXT ("AE_AML_REGION_LIMIT",           "Tried to access beyond the end
302     EXCEP_TXT ("AE_AML_BUFFER_LIMIT",           "Tried to access beyond the end
303     EXCEP_TXT ("AE_AML_PACKAGE_LIMIT",          "Tried to access beyond the end
304     EXCEP_TXT ("AE_AML_DIVIDE_BY_ZERO",         "During execution of AML Divide
305     EXCEP_TXT ("AE_AML_BAD_NAME",               "An ACPI name contains invalid c
306     EXCEP_TXT ("AE_AML_NAME_NOT_FOUND",         "Could not resolve a named refer
307     EXCEP_TXT ("AE_AML_INTERNAL",               "An internal error within the in
308     EXCEP_TXT ("AE_AML_INVALID_SPACE_ID",       "An Operation Region SpaceID is
309     EXCEP_TXT ("AE_AML_STRING_LIMIT",           "String is longer than 200 chara
310     EXCEP_TXT ("AE_AML_NO_RETURN_VALUE",        "A method did not return a requi
311     EXCEP_TXT ("AE_AML_METHOD_LIMIT",           "A control method reached the ma
312     EXCEP_TXT ("AE_AML_NOT_OWNER",              "A thread tried to release a mut
313     EXCEP_TXT ("AE_AML_MUTEX_ORDER",            "Mutex SyncLevel release mismatc
314     EXCEP_TXT ("AE_AML_MUTEX_NOT_ACQUIRED",     "Attempt to release a mutex that
315     EXCEP_TXT ("AE_AML_INVALID_RESOURCE_TYPE",  "Invalid resource type in resour
316     EXCEP_TXT ("AE_AML_INVALID_INDEX",          "Invalid Argx or Localx (x too l
317     EXCEP_TXT ("AE_AML_REGISTER_LIMIT",         "Bank value or Index value beyon
318     EXCEP_TXT ("AE_AML_NO_WHILE",               "Break or Continue without a Whi
319     EXCEP_TXT ("AE_AML_ALIGNMENT",              "Non-aligned memory transfer on
320     EXCEP_TXT ("AE_AML_NO_RESOURCE_END_TAG",    "No End Tag in a resource list")
321     EXCEP_TXT ("AE_AML_BAD_RESOURCE_VALUE",     "Invalid value of a resource ele
322     EXCEP_TXT ("AE_AML_CIRCULAR_REFERENCE",     "Two references refer to each ot
323     EXCEP_TXT ("AE_AML_BAD_RESOURCE_LENGTH",    "The length of a Resource Descri
324     EXCEP_TXT ("AE_AML_ILLEGAL_ADDRESS",        "A memory, I/O, or PCI configura
```



```
325     EXCEP_TXT ("AE_AML_INFINITE_LOOP",          "An apparent infinite AML While
326 };

328 static const ACPI_EXCEPTION_INFO    AcpiGbl_ExceptionNames_Ctrl[] =
329 {
330     EXCEP_TXT (NULL, NULL),
331     EXCEP_TXT ("AE_CTRL_RETURN_VALUE",          "A Method returned a value"),
332     EXCEP_TXT ("AE_CTRL_PENDING",               "Method is calling another metho
333     EXCEP_TXT ("AE_CTRL_TERMINATE",             "Terminate the executing method"
334     EXCEP_TXT ("AE_CTRL_TRUE",                  "An If or While predicate result
335     EXCEP_TXT ("AE_CTRL_FALSE",                 "An If or While predicate result
336     EXCEP_TXT ("AE_CTRL_DEPTH",                 "Maximum search depth has been r
337     EXCEP_TXT ("AE_CTRL_END",                   "An If or While predicate is fal
338     EXCEP_TXT ("AE_CTRL_TRANSFER",              "Transfer control to called meth
339     EXCEP_TXT ("AE_CTRL_BREAK",                 "A Break has been executed"),
340     EXCEP_TXT ("AE_CTRL_CONTINUE",              "A Continue has been executed"),
341     EXCEP_TXT ("AE_CTRL_SKIP",                  "Not currently used"),
342     EXCEP_TXT ("AE_CTRL_PARSE_CONTINUE",        "Used to skip over bad opcodes")
343     EXCEP_TXT ("AE_CTRL_PARSE_PENDING",         "Used to implement AML While loo
344 };

346 #endif /* EXCEPTION_TABLE */

348 #endif /* __ACEXCEP_H__ */
```

```
**********************************************************
   20926 Thu Dec 26 13:49:48 2013
new/usr/src/common/acpica/include/acglobal.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acglobal.h - Declarations for global variables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACGLOBAL_H__
#define __ACGLOBAL_H__


/*
 * Ensure that the globals are actually defined and initialized only once.
 *
 * The use of these macros allows a single list of globals (here) in order
 * to simplify maintenance of the code.
 */
#ifdef DEFINE_ACPI_GLOBALS
#define ACPI_EXTERN
#define ACPI_INIT_GLOBAL(a,b) a=b
#else
#define ACPI_EXTERN extern
```



```
#define ACPI_INIT_GLOBAL(a,b) a
#endif


#ifdef DEFINE_ACPI_GLOBALS

/* Public globals, available from outside ACPICA subsystem */

/******************************************************************************
 *
 * Runtime configuration (static defaults that can be overriden at runtime)
 *
 *****************************************************************************/

/*
 * Enable "slack" in the AML interpreter?  Default is FALSE, and the
 * interpreter strictly follows the ACPI specification. Setting to TRUE
 * allows the interpreter to ignore certain errors and/or bad AML constructs.
 *
 * Currently, these features are enabled by this flag:
 *
 * 1) Allow "implicit return" of last value in a control method
 * 2) Allow access beyond the end of an operation region
 * 3) Allow access to uninitialized locals/args (auto-init to integer 0)
 * 4) Allow ANY object type to be a source operand for the Store() operator
 * 5) Allow unresolved references (invalid target name) in package objects
 * 6) Enable warning messages for behavior that is not ACPI spec compliant
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_EnableInterpreterSlack, FALSE);

/*
 * Automatically serialize ALL control methods? Default is FALSE, meaning
 * to use the Serialized/NotSerialized method flags on a per method basis.
 * Only change this if the ASL code is poorly written and cannot handle
 * reentrancy even though methods are marked "NotSerialized".
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_AllMethodsSerialized, FALSE);

/*
 * Create the predefined _OSI method in the namespace? Default is TRUE
 * because ACPI CA is fully compatible with other ACPI implementations.
 * Changing this will revert ACPI CA (and machine ASL) to pre-OSI behavior.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_CreateOsiMethod, TRUE);

/*
 * Optionally use default values for the ACPI register widths. Set this to
 * TRUE to use the defaults, if an FADT contains incorrect widths/lengths.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_UseDefaultRegisterWidths, TRUE);

/*
 * Optionally enable output from the AML Debug Object.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_EnableAmlDebugObject, FALSE);

/*
 * Optionally copy the entire DSDT to local memory (instead of simply
 * mapping it.) There are some BIOSs that corrupt or replace the original
 * DSDT, creating the need for this option. Default is FALSE, do not copy
 * the DSDT.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_CopyDsdtLocally, FALSE);

/*
 * Optionally ignore an XSDT if present and use the RSDT instead.
```

```
 * Although the ACPI specification requires that an XSDT be used instead
 * of the RSDT, the XSDT has been found to be corrupt or ill-formed on
 * some machines. Default behavior is to use the XSDT if present.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_DoNotUseXsdt, FALSE);


/*
 * Optionally use 32-bit FADT addresses if and when there is a conflict
 * (address mismatch) between the 32-bit and 64-bit versions of the
 * address. Although ACPICA adheres to the ACPI specification which
 * requires the use of the corresponding 64-bit address if it is non-zero,
 * some machines have been found to have a corrupted non-zero 64-bit
 * address. Default is FALSE, do not favor the 32-bit addresses.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_Use32BitFadtAddresses, FALSE);

/*
 * Optionally truncate I/O addresses to 16 bits. Provides compatibility
 * with other ACPI implementations. NOTE: During ACPICA initialization,
 * this value is set to TRUE if any Windows OSI strings have been
 * requested by the BIOS.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_TruncateIoAddresses, FALSE);

/*
 * Disable runtime checking and repair of values returned by control methods.
 * Use only if the repair is causing a problem on a particular machine.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_DisableAutoRepair, FALSE);

/*
 * Optionally do not load any SSDTs from the RSDT/XSDT during initialization.
 * This can be useful for debugging ACPI problems on some machines.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_DisableSsdtTableLoad, FALSE);

/*
 * We keep track of the latest version of Windows that has been requested by
 * the BIOS.
 */
UINT8       ACPI_INIT_GLOBAL (AcpiGbl_OsiData, 0);


/* AcpiGbl_FADT is a local copy of the FADT, converted to a common format. */

ACPI_TABLE_FADT             AcpiGbl_FADT;
UINT32                      AcpiCurrentGpeCount;
UINT32                      AcpiGbl_TraceFlags;
ACPI_NAME                   AcpiGbl_TraceMethodName;
BOOLEAN                     AcpiGbl_SystemAwakeAndRunning;

/*
 * ACPI 5.0 introduces the concept of a "reduced hardware platform", meaning
 * that the ACPI hardware is no longer required. A flag in the FADT indicates
 * a reduced HW machine, and that flag is duplicated here for convenience.
 */
BOOLEAN                     AcpiGbl_ReducedHardware;
#endif

#endif /* DEFINE_ACPI_GLOBALS */

/* Do not disassemble buffers to resource descriptors */

ACPI_EXTERN UINT8       ACPI_INIT_GLOBAL (AcpiGbl_NoResourceDisassembly, FALSE);
```



```
/*****************************************************************************
 *
 * ACPI Table globals
 *
 ****************************************************************************/

/*
 * AcpiGbl_RootTableList is the master list of ACPI tables that were
 * found in the RSDT/XSDT.
 */
ACPI_EXTERN ACPI_TABLE_LIST             AcpiGbl_RootTableList;

#if (!ACPI_REDUCED_HARDWARE)
ACPI_EXTERN ACPI_TABLE_FACS            *AcpiGbl_FACS;

#endif /* !ACPI_REDUCED_HARDWARE */

/* These addresses are calculated from the FADT Event Block addresses */

ACPI_EXTERN ACPI_GENERIC_ADDRESS        AcpiGbl_XPm1aStatus;
ACPI_EXTERN ACPI_GENERIC_ADDRESS        AcpiGbl_XPm1aEnable;

ACPI_EXTERN ACPI_GENERIC_ADDRESS        AcpiGbl_XPm1bStatus;
ACPI_EXTERN ACPI_GENERIC_ADDRESS        AcpiGbl_XPm1bEnable;

/* DSDT information. Used to check for DSDT corruption */

ACPI_EXTERN ACPI_TABLE_HEADER          *AcpiGbl_DSDT;
ACPI_EXTERN ACPI_TABLE_HEADER           AcpiGbl_OriginalDsdtHeader;

/*
 * Handle both ACPI 1.0 and ACPI 2.0 Integer widths. The integer width is
 * determined by the revision of the DSDT: If the DSDT revision is less than
 * 2, use only the lower 32 bits of the internal 64-bit Integer.
 */
ACPI_EXTERN UINT8                       AcpiGbl_IntegerBitWidth;
ACPI_EXTERN UINT8                       AcpiGbl_IntegerByteWidth;
ACPI_EXTERN UINT8                       AcpiGbl_IntegerNybbleWidth;


/*****************************************************************************
 *
 * Mutual exclusion within ACPICA subsystem
 * Mutual exlusion within ACPICA subsystem
 *
 ****************************************************************************/

/*
 * Predefined mutex objects. This array contains the
 * actual OS mutex handles, indexed by the local ACPI_MUTEX_HANDLEs.
 * (The table maps local handles to the real OS handles)
 */
ACPI_EXTERN ACPI_MUTEX_INFO             AcpiGbl_MutexInfo[ACPI_NUM_MUTEX];

/*
 * Global lock mutex is an actual AML mutex object
 * Global lock semaphore works in conjunction with the actual global lock
 * Global lock spinlock is used for "pending" handshake
 */
ACPI_EXTERN ACPI_OPERAND_OBJECT        *AcpiGbl_GlobalLockMutex;
ACPI_EXTERN ACPI_SEMAPHORE              AcpiGbl_GlobalLockSemaphore;
ACPI_EXTERN ACPI_SPINLOCK               AcpiGbl_GlobalLockPendingLock;
ACPI_EXTERN UINT16                      AcpiGbl_GlobalLockHandle;
ACPI_EXTERN BOOLEAN                     AcpiGbl_GlobalLockAcquired;
ACPI_EXTERN BOOLEAN                     AcpiGbl_GlobalLockPresent;
ACPI_EXTERN BOOLEAN                     AcpiGbl_GlobalLockPending;
```

```
256 /*
257  * Spinlocks are used for interfaces that can be possibly called at
258  * interrupt level
259  */
260 ACPI_EXTERN ACPI_SPINLOCK               AcpiGbl_GpeLock;      /* For GPE data st
261 ACPI_EXTERN ACPI_SPINLOCK               AcpiGbl_HardwareLock; /* For ACPI H/W ex
262 ACPI_EXTERN ACPI_SPINLOCK               AcpiGbl_ReferenceCountLock;

264 /* Mutex for _OSI support */

266 ACPI_EXTERN ACPI_MUTEX                  AcpiGbl_OsiMutex;

268 /* Reader/Writer lock is used for namespace walk and dynamic table unload */

270 ACPI_EXTERN ACPI_RW_LOCK                AcpiGbl_NamespaceRwLock;


273 /*****************************************************************************
274  *
275  * Miscellaneous globals
276  *
277  ****************************************************************************/

279 /* Object caches */

281 ACPI_EXTERN ACPI_CACHE_T                *AcpiGbl_NamespaceCache;
282 ACPI_EXTERN ACPI_CACHE_T                *AcpiGbl_StateCache;
283 ACPI_EXTERN ACPI_CACHE_T                *AcpiGbl_PsNodeCache;
284 ACPI_EXTERN ACPI_CACHE_T                *AcpiGbl_PsNodeExtCache;
285 ACPI_EXTERN ACPI_CACHE_T                *AcpiGbl_OperandCache;

287 /* Global handlers */

289 ACPI_EXTERN ACPI_GLOBAL_NOTIFY_HANDLER  AcpiGbl_GlobalNotify[2];
236 ACPI_EXTERN ACPI_OBJECT_NOTIFY_HANDLER  AcpiGbl_DeviceNotify;
237 ACPI_EXTERN ACPI_OBJECT_NOTIFY_HANDLER  AcpiGbl_SystemNotify;
290 ACPI_EXTERN ACPI_EXCEPTION_HANDLER      AcpiGbl_ExceptionHandler;
291 ACPI_EXTERN ACPI_INIT_HANDLER           AcpiGbl_InitHandler;
292 ACPI_EXTERN ACPI_TABLE_HANDLER          AcpiGbl_TableHandler;
293 ACPI_EXTERN void                        *AcpiGbl_TableHandlerContext;
294 ACPI_EXTERN ACPI_WALK_STATE             *AcpiGbl_BreakpointWalk;
295 ACPI_EXTERN ACPI_INTERFACE_HANDLER      AcpiGbl_InterfaceHandler;
296 ACPI_EXTERN ACPI_SCI_HANDLER_INFO       *AcpiGbl_SciHandlerList;

298 /* Owner ID support */

300 ACPI_EXTERN UINT32                      AcpiGbl_OwnerIdMask[ACPI_NUM_OWNERID_MAS
301 ACPI_EXTERN UINT8                       AcpiGbl_LastOwnerIdIndex;
302 ACPI_EXTERN UINT8                       AcpiGbl_NextOwnerIdOffset;

304 /* Initialization sequencing */

306 ACPI_EXTERN BOOLEAN                     AcpiGbl_RegMethodsExecuted;

308 /* Misc */

310 ACPI_EXTERN UINT32                      AcpiGbl_OriginalMode;
311 ACPI_EXTERN UINT32                      AcpiGbl_RsdpOriginalLocation;
312 ACPI_EXTERN UINT32                      AcpiGbl_NsLookupCount;
313 ACPI_EXTERN UINT32                      AcpiGbl_PsFindCount;
314 ACPI_EXTERN UINT16                      AcpiGbl_Pm1EnableRegisterSave;
315 ACPI_EXTERN UINT8                       AcpiGbl_DebuggerConfiguration;
316 ACPI_EXTERN BOOLEAN                     AcpiGbl_StepToNextCall;
317 ACPI_EXTERN BOOLEAN                     AcpiGbl_AcpiHardwarePresent;
318 ACPI_EXTERN BOOLEAN                     AcpiGbl_EventsInitialized;
```



```
266 ACPI_EXTERN UINT8                       AcpiGbl_OsiData;
319 ACPI_EXTERN ACPI_INTERFACE_INFO         *AcpiGbl_SupportedInterfaces;
320 ACPI_EXTERN ACPI_ADDRESS_RANGE          *AcpiGbl_AddressRangeList[ACPI_ADDRESS_RA


322 #ifndef DEFINE_ACPI_GLOBALS

272 /* Exception codes */

274 extern char const                       *AcpiGbl_ExceptionNames_Env[];
275 extern char const                       *AcpiGbl_ExceptionNames_Pgm[];
276 extern char const                       *AcpiGbl_ExceptionNames_Tbl[];
277 extern char const                       *AcpiGbl_ExceptionNames_Aml[];
278 extern char const                       *AcpiGbl_ExceptionNames_Ctrl[];

324 /* Other miscellaneous */

326 extern BOOLEAN                          AcpiGbl_Shutdown;
327 extern UINT32                           AcpiGbl_StartupFlags;
328 extern const char                       *AcpiGbl_SleepStateNames[ACPI_S_STATE_COU
329 extern const char                       *AcpiGbl_LowestDstateNames[ACPI_NUM_SxW_M
330 extern const char                       *AcpiGbl_HighestDstateNames[ACPI_NUM_SxD_
331 extern const ACPI_OPCODE_INFO           AcpiGbl_AmlOpInfo[AML_NUM_OPCODES];
332 extern const char                       *AcpiGbl_RegionTypes[ACPI_NUM_PREDEFINED_
333 #endif


336 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

338 /* Lists for tracking memory allocations */

340 ACPI_EXTERN ACPI_MEMORY_LIST            *AcpiGbl_GlobalList;
341 ACPI_EXTERN ACPI_MEMORY_LIST            *AcpiGbl_NsNodeList;
342 ACPI_EXTERN BOOLEAN                     AcpiGbl_DisplayFinalMemStats;
343 ACPI_EXTERN BOOLEAN                     AcpiGbl_DisableMemTracking;
344 #endif


347 /*****************************************************************************
348  *
349  * Namespace globals
350  *
351  ****************************************************************************/

353 #if !defined (ACPI_NO_METHOD_EXECUTION) || defined (ACPI_CONSTANT_EVAL_ONLY)
354 #define NUM_PREDEFINED_NAMES            10
355 #else
356 #define NUM_PREDEFINED_NAMES            9
357 #endif

359 ACPI_EXTERN ACPI_NAMESPACE_NODE         AcpiGbl_RootNodeStruct;
360 ACPI_EXTERN ACPI_NAMESPACE_NODE         *AcpiGbl_RootNode;
361 ACPI_EXTERN ACPI_NAMESPACE_NODE         *AcpiGbl_FadtGpeDevice;
362 ACPI_EXTERN ACPI_OPERAND_OBJECT         *AcpiGbl_ModuleCodeList;


365 extern const UINT8                      AcpiGbl_NsProperties [ACPI_NUM_NS_TYPES]
366 extern const ACPI_PREDEFINED_NAMES      AcpiGbl_PreDefinedNames [NUM_PREDEFINED_

368 #ifdef ACPI_DEBUG_OUTPUT
369 ACPI_EXTERN UINT32                      AcpiGbl_CurrentNodeCount;
370 ACPI_EXTERN UINT32                      AcpiGbl_CurrentNodeSize;
371 ACPI_EXTERN UINT32                      AcpiGbl_MaxConcurrentNodeCount;
372 ACPI_EXTERN ACPI_SIZE                   *AcpiGbl_EntryStackPointer;
373 ACPI_EXTERN ACPI_SIZE                   *AcpiGbl_LowestStackPointer;
374 ACPI_EXTERN UINT32                      AcpiGbl_DeepestNesting;
```

```
375 #endif


378 /*****************************************************************************
379  *
380  * Interpreter globals
381  *
382  ****************************************************************************/


384 ACPI_EXTERN ACPI_THREAD_STATE          *AcpiGbl_CurrentWalkList;

386 /* Control method single step flag */

388 ACPI_EXTERN UINT8                       AcpiGbl_CmSingleStep;


391 /*****************************************************************************
392  *
393  * Hardware globals
394  *
395  ****************************************************************************/

397 extern      ACPI_BIT_REGISTER_INFO      AcpiGbl_BitRegisterInfo[ACPI_NUM_BITREG]
398 ACPI_EXTERN UINT8                       AcpiGbl_SleepTypeA;
399 ACPI_EXTERN UINT8                       AcpiGbl_SleepTypeB;


402 /*****************************************************************************
403  *
404  * Event and GPE globals
405  *
406  ****************************************************************************/

408 #if (!ACPI_REDUCED_HARDWARE)

410 ACPI_EXTERN UINT8                       AcpiGbl_AllGpesInitialized;
411 ACPI_EXTERN ACPI_GPE_XRUPT_INFO        *AcpiGbl_GpeXruptListHead;
412 ACPI_EXTERN ACPI_GPE_BLOCK_INFO        *AcpiGbl_GpeFadtBlocks[ACPI_MAX_GPE_BLOCK
413 ACPI_EXTERN ACPI_GBL_EVENT_HANDLER      AcpiGbl_GlobalEventHandler;
414 ACPI_EXTERN void                       *AcpiGbl_GlobalEventHandlerContext;
415 ACPI_EXTERN ACPI_FIXED_EVENT_HANDLER    AcpiGbl_FixedEventHandlers[ACPI_NUM_FIXE
416 extern      ACPI_FIXED_EVENT_INFO       AcpiGbl_FixedEventInfo[ACPI_NUM_FIXED_EV

418 #endif /* !ACPI_REDUCED_HARDWARE */

420 /*****************************************************************************
421  *
422  * Debug support
423  *
424  ****************************************************************************/

426 /* Procedure nesting level for debug output */

428 extern      UINT32                      AcpiGbl_NestingLevel;

430 /* Event counters */

432 ACPI_EXTERN UINT32                      AcpiMethodCount;
433 ACPI_EXTERN UINT32                      AcpiGpeCount;
434 ACPI_EXTERN UINT32                      AcpiSciCount;
435 ACPI_EXTERN UINT32                      AcpiFixedEventCount[ACPI_NUM_FIXED_EVENT

437 /* Support for dynamic control method tracing mechanism */

439 ACPI_EXTERN UINT32                      AcpiGbl_OriginalDbgLevel;
```



```
440 ACPI_EXTERN UINT32                      AcpiGbl_OriginalDbgLayer;
441 ACPI_EXTERN UINT32                      AcpiGbl_TraceDbgLevel;
442 ACPI_EXTERN UINT32                      AcpiGbl_TraceDbgLayer;


445 /*****************************************************************************
446  *
447  * Debugger and Disassembler globals
401  * Debugger globals
448  *
449  ****************************************************************************/

451 ACPI_EXTERN UINT8                       AcpiGbl_DbOutputFlags;

453 #ifdef ACPI_DISASSEMBLER

455 ACPI_EXTERN BOOLEAN                     ACPI_INIT_GLOBAL (AcpiGbl_IgnoreNoopOper

457 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_disasm;
458 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_verbose;
459 ACPI_EXTERN BOOLEAN                     AcpiGbl_NumExternalMethods;
460 ACPI_EXTERN UINT32                      AcpiGbl_ResolvedExternalMethods;
461 ACPI_EXTERN ACPI_EXTERNAL_LIST         *AcpiGbl_ExternalList;
462 ACPI_EXTERN ACPI_EXTERNAL_FILE         *AcpiGbl_ExternalFileList;
463 #endif


466 #ifdef ACPI_DEBUGGER

468 extern      BOOLEAN                     AcpiGbl_MethodExecuting;
469 extern      BOOLEAN                     AcpiGbl_AbortMethod;
470 extern      BOOLEAN                     AcpiGbl_DbTerminateThreads;

472 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_tables;
473 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_stats;
474 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_ini_methods;
475 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOpt_NoRegionSupport;

427 ACPI_EXTERN char                       *AcpiGbl_DbArgs[ACPI_DEBUGGER_MAX_ARGS];
428 ACPI_EXTERN ACPI_OBJECT_TYPE            AcpiGbl_DbArgTypes[ACPI_DEBUGGER_MAX_ARG
429 ACPI_EXTERN char                        AcpiGbl_DbLineBuf[ACPI_DB_LINE_BUFFER_SI
430 ACPI_EXTERN char                        AcpiGbl_DbParsedBuf[ACPI_DB_LINE_BUFFER_
431 ACPI_EXTERN char                        AcpiGbl_DbScopeBuf[80];
432 ACPI_EXTERN char                        AcpiGbl_DbDebugFilename[80];
476 ACPI_EXTERN BOOLEAN                     AcpiGbl_DbOutputToFile;
477 ACPI_EXTERN char                       *AcpiGbl_DbBuffer;
478 ACPI_EXTERN char                       *AcpiGbl_DbFilename;
479 ACPI_EXTERN UINT32                      AcpiGbl_DbDebugLevel;
480 ACPI_EXTERN UINT32                      AcpiGbl_DbConsoleDebugLevel;
481 ACPI_EXTERN ACPI_NAMESPACE_NODE        *AcpiGbl_DbScopeNode;

483 ACPI_EXTERN char                       *AcpiGbl_DbArgs[ACPI_DEBUGGER_MAX_ARGS];
484 ACPI_EXTERN ACPI_OBJECT_TYPE            AcpiGbl_DbArgTypes[ACPI_DEBUGGER_MAX_ARG

486 /* These buffers should all be the same size */

488 ACPI_EXTERN char                        AcpiGbl_DbLineBuf[ACPI_DB_LINE_BUFFER_SI
489 ACPI_EXTERN char                        AcpiGbl_DbParsedBuf[ACPI_DB_LINE_BUFFER_
490 ACPI_EXTERN char                        AcpiGbl_DbScopeBuf[ACPI_DB_LINE_BUFFER_S
491 ACPI_EXTERN char                        AcpiGbl_DbDebugFilename[ACPI_DB_LINE_BUF

493 /*
494  * Statistic globals
495  */
496 ACPI_EXTERN UINT16                      AcpiGbl_ObjTypeCount[ACPI_TYPE_NS_NODE_M
497 ACPI_EXTERN UINT16                      AcpiGbl_NodeTypeCount[ACPI_TYPE_NS_NODE_
```

```
498 ACPI_EXTERN UINT16                  AcpiGbl_ObjTypeCountMisc;
499 ACPI_EXTERN UINT16                  AcpiGbl_NodeTypeCountMisc;
500 ACPI_EXTERN UINT32                  AcpiGbl_NumNodes;
501 ACPI_EXTERN UINT32                  AcpiGbl_NumObjects;


504 ACPI_EXTERN UINT32                  AcpiGbl_SizeOfParseTree;
505 ACPI_EXTERN UINT32                  AcpiGbl_SizeOfMethodTrees;
506 ACPI_EXTERN UINT32                  AcpiGbl_SizeOfNodeEntries;
507 ACPI_EXTERN UINT32                  AcpiGbl_SizeOfAcpiObjects;

509 #endif /* ACPI_DEBUGGER */


512 /*****************************************************************************
513  *
514  * Application globals
515  *
516  ****************************************************************************/

518 #ifdef ACPI_APPLICATION

520 ACPI_FILE   ACPI_INIT_GLOBAL (AcpiGbl_DebugFile, NULL);

522 #endif /* ACPI_APPLICATION */


525 /*****************************************************************************
526  *
527  * Info/help support
528  *
529  ****************************************************************************/

531 extern const AH_PREDEFINED_NAME     AslPredefinedInfo[];


534 #endif /* __ACGLOBAL_H__ */
```

```
**********************************************************
    5682 Thu Dec 26 13:49:49 2013
new/usr/src/common/acpica/include/achware.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Name: achware.h -- hardware specific interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACHWARE_H__
#define __ACHWARE_H__


/* Values for the _SST predefined method */

#define ACPI_SST_INDICATOR_OFF  0
#define ACPI_SST_WORKING        1
#define ACPI_SST_WAKING         2
#define ACPI_SST_SLEEPING       3
#define ACPI_SST_SLEEP_CONTEXT  4


/*
 * hwacpi - high level functions
 */
```



```
ACPI_STATUS
AcpiHwSetMode (
    UINT32                  Mode);

UINT32
AcpiHwGetMode (
    void);


/*
 * hwregs - ACPI Register I/O
 */
ACPI_STATUS
AcpiHwValidateRegister (
    ACPI_GENERIC_ADDRESS    *Reg,
    UINT8                   MaxBitWidth,
    UINT64                  *Address);

ACPI_STATUS
AcpiHwRead (
    UINT32                  *Value,
    ACPI_GENERIC_ADDRESS    *Reg);

ACPI_STATUS
AcpiHwWrite (
    UINT32                  Value,
    ACPI_GENERIC_ADDRESS    *Reg);

ACPI_BIT_REGISTER_INFO *
AcpiHwGetBitRegisterInfo (
    UINT32                  RegisterId);

ACPI_STATUS
AcpiHwWritePm1Control (
    UINT32                  Pm1aControl,
    UINT32                  Pm1bControl);

ACPI_STATUS
AcpiHwRegisterRead (
    UINT32                  RegisterId,
    UINT32                  *ReturnValue);

ACPI_STATUS
AcpiHwRegisterWrite (
    UINT32                  RegisterId,
    UINT32                  Value);

ACPI_STATUS
AcpiHwClearAcpiStatus (
    void);


/*
 * hwsleep - sleep/wake support (Legacy sleep registers)
 */
ACPI_STATUS
AcpiHwLegacySleep (
    UINT8                   SleepState);

ACPI_STATUS
AcpiHwLegacyWakePrep (
    UINT8                   SleepState);

ACPI_STATUS
AcpiHwLegacyWake (
    UINT8                   SleepState);
```

```
128 /*
129  * hwesleep - sleep/wake support (Extended FADT-V5 sleep registers)
130  */
131 void
132 AcpiHwExecuteSleepMethod (
133     char                    *MethodName,
134     UINT32                  IntegerArgument);

136 ACPI_STATUS
137 AcpiHwExtendedSleep (
138     UINT8                   SleepState);

140 ACPI_STATUS
141 AcpiHwExtendedWakePrep (
142     UINT8                   SleepState);

144 ACPI_STATUS
145 AcpiHwExtendedWake (
146     UINT8                   SleepState);


149 /*
150  * hwvalid - Port I/O with validation
151  */
152 ACPI_STATUS
153 AcpiHwReadPort (
154     ACPI_IO_ADDRESS         Address,
155     UINT32                  *Value,
156     UINT32                  Width);

158 ACPI_STATUS
159 AcpiHwWritePort (
160     ACPI_IO_ADDRESS         Address,
161     UINT32                  Value,
162     UINT32                  Width);


165 /*
166  * hwgpe - GPE support
167  */
168 UINT32
169 AcpiHwGetGpeRegisterBit (
170     ACPI_GPE_EVENT_INFO     *GpeEventInfo);
133     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
134     ACPI_GPE_REGISTER_INFO  *GpeRegisterInfo);

172 ACPI_STATUS
173 AcpiHwLowSetGpe (
174     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
175     UINT32                  Action);

177 ACPI_STATUS
178 AcpiHwDisableGpeBlock (
179     ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
180     ACPI_GPE_BLOCK_INFO     *GpeBlock,
181     void                    *Context);

183 ACPI_STATUS
184 AcpiHwClearGpe (
185     ACPI_GPE_EVENT_INFO     *GpeEventInfo);

187 ACPI_STATUS
188 AcpiHwClearGpeBlock (
189     ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
```



```
190     ACPI_GPE_BLOCK_INFO     *GpeBlock,
191     void                    *Context);

193 ACPI_STATUS
194 AcpiHwGetGpeStatus (
195     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
196     ACPI_EVENT_STATUS       *EventStatus);

198 ACPI_STATUS
199 AcpiHwDisableAllGpes (
200     void);

202 ACPI_STATUS
203 AcpiHwEnableAllRuntimeGpes (
204     void);

206 ACPI_STATUS
207 AcpiHwEnableAllWakeupGpes (
208     void);

210 ACPI_STATUS
211 AcpiHwEnableRuntimeGpeBlock (
212     ACPI_GPE_XRUPT_INFO     *GpeXruptInfo,
213     ACPI_GPE_BLOCK_INFO     *GpeBlock,
214     void                    *Context);


217 /*
218  * hwpci - PCI configuration support
219  */
220 ACPI_STATUS
221 AcpiHwDerivePciId (
222     ACPI_PCI_ID             *PciId,
223     ACPI_HANDLE             RootPciDevice,
224     ACPI_HANDLE             PciRegion);


191 /*
192  * hwtimer - ACPI Timer prototypes
193  */
194 ACPI_STATUS
195 AcpiGetTimerResolution (
196     UINT32                  *Resolution);

198 ACPI_STATUS
199 AcpiGetTimer (
200     UINT32                  *Ticks);

202 ACPI_STATUS
203 AcpiGetTimerDuration (
204     UINT32                  StartTicks,
205     UINT32                  EndTicks,
206     UINT32                  *TimeElapsed);


227 #endif /* __ACHWARE_H__ */
```

```
**********************************************************
   18202 Thu Dec 26 13:49:49 2013
new/usr/src/common/acpica/include/acinterp.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acinterp.h - Interpreter subcomponent prototypes and defines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACINTERP_H__
  45 #define __ACINTERP_H__


  48 #define ACPI_WALK_OPERANDS          (&(WalkState->Operands [WalkState->NumOperan

  50 /* Macros for tables used for debug output */

  52 #define ACPI_EXD_OFFSET(f)          (UINT8) ACPI_OFFSET (ACPI_OPERAND_OBJECT,f)
  53 #define ACPI_EXD_NSOFFSET(f)        (UINT8) ACPI_OFFSET (ACPI_NAMESPACE_NODE,f)
  54 #define ACPI_EXD_TABLE_SIZE(name)   (sizeof(name) / sizeof (ACPI_EXDUMP_INFO))

  56 /*
  57  * If possible, pack the following structures to byte alignment, since we
  58  * don't care about performance for debug output. Two cases where we cannot
  59  * pack the structures:
```



```
  60  *
  61  * 1) Hardware does not support misaligned memory transfers
  62  * 2) Compiler does not support pointers within packed structures
  63  */
  64 #if (!defined(ACPI_MISALIGNMENT_NOT_SUPPORTED) && !defined(ACPI_PACKED_POINTERS_
  65 #pragma pack(1)
  66 #endif

  68 typedef const struct acpi_exdump_info
  69 {
  70     UINT8                   Opcode;
  71     UINT8                   Offset;
  72     char                    *Name;

  74 } ACPI_EXDUMP_INFO;

  76 /* Values for the Opcode field above */

  78 #define ACPI_EXD_INIT                   0
  79 #define ACPI_EXD_TYPE                   1
  80 #define ACPI_EXD_UINT8                  2
  81 #define ACPI_EXD_UINT16                 3
  82 #define ACPI_EXD_UINT32                 4
  83 #define ACPI_EXD_UINT64                 5
  84 #define ACPI_EXD_LITERAL                6
  85 #define ACPI_EXD_POINTER                7
  86 #define ACPI_EXD_ADDRESS                8
  87 #define ACPI_EXD_STRING                 9
  88 #define ACPI_EXD_BUFFER                 10
  89 #define ACPI_EXD_PACKAGE                11
  90 #define ACPI_EXD_FIELD                  12
  91 #define ACPI_EXD_REFERENCE              13

  93 /* restore default alignment */

  95 #pragma pack()


  98 /*
  99  * exconvrt - object conversion
 100  */
 101 ACPI_STATUS
 102 AcpiExConvertToInteger (
 103     ACPI_OPERAND_OBJECT     *ObjDesc,
 104     ACPI_OPERAND_OBJECT     **ResultDesc,
 105     UINT32                  Flags);

 107 ACPI_STATUS
 108 AcpiExConvertToBuffer (
 109     ACPI_OPERAND_OBJECT     *ObjDesc,
 110     ACPI_OPERAND_OBJECT     **ResultDesc);

 112 ACPI_STATUS
 113 AcpiExConvertToString (
 114     ACPI_OPERAND_OBJECT     *ObjDesc,
 115     ACPI_OPERAND_OBJECT     **ResultDesc,
 116     UINT32                  Type);

 118 /* Types for ->String conversion */

 120 #define ACPI_EXPLICIT_BYTE_COPY         0x00000000
 121 #define ACPI_EXPLICIT_CONVERT_HEX       0x00000001
 122 #define ACPI_IMPLICIT_CONVERT_HEX       0x00000002
 123 #define ACPI_EXPLICIT_CONVERT_DECIMAL   0x00000003

 125 ACPI_STATUS
```

```
126 AcpiExConvertToTargetType (
127     ACPI_OBJECT_TYPE        DestinationType,
128     ACPI_OPERAND_OBJECT     *SourceDesc,
129     ACPI_OPERAND_OBJECT     **ResultDesc,
130     ACPI_WALK_STATE         *WalkState);


133 /*
134  * exdebug - AML debug object
135  */
136 void
137 AcpiExDoDebugObject (
138     ACPI_OPERAND_OBJECT     *SourceDesc,
139     UINT32                  Level,
140     UINT32                  Index);


143 /*
144  * exfield - ACPI AML (p-code) execution - field manipulation
145  */
146 ACPI_STATUS
147 AcpiExCommonBufferSetup (
148     ACPI_OPERAND_OBJECT     *ObjDesc,
149     UINT32                  BufferLength,
150     UINT32                  *DatumCount);

152 ACPI_STATUS
153 AcpiExWriteWithUpdateRule (
154     ACPI_OPERAND_OBJECT     *ObjDesc,
155     UINT64                  Mask,
156     UINT64                  FieldValue,
157     UINT32                  FieldDatumByteOffset);

159 void
160 AcpiExGetBufferDatum(
161     UINT64                  *Datum,
162     void                    *Buffer,
163     UINT32                  BufferLength,
164     UINT32                  ByteGranularity,
165     UINT32                  BufferOffset);

167 void
168 AcpiExSetBufferDatum (
169     UINT64                  MergedDatum,
170     void                    *Buffer,
171     UINT32                  BufferLength,
172     UINT32                  ByteGranularity,
173     UINT32                  BufferOffset);

175 ACPI_STATUS
176 AcpiExReadDataFromField (
177     ACPI_WALK_STATE         *WalkState,
178     ACPI_OPERAND_OBJECT     *ObjDesc,
179     ACPI_OPERAND_OBJECT     **RetBufferDesc);

181 ACPI_STATUS
182 AcpiExWriteDataToField (
183     ACPI_OPERAND_OBJECT     *SourceDesc,
184     ACPI_OPERAND_OBJECT     *ObjDesc,
185     ACPI_OPERAND_OBJECT     **ResultDesc);


188 /*
189  * exfldio - low level field I/O
190  */
191 ACPI_STATUS
```



```
192 AcpiExExtractFromField (
193     ACPI_OPERAND_OBJECT     *ObjDesc,
194     void                    *Buffer,
195     UINT32                  BufferLength);

197 ACPI_STATUS
198 AcpiExInsertIntoField (
199     ACPI_OPERAND_OBJECT     *ObjDesc,
200     void                    *Buffer,
201     UINT32                  BufferLength);

203 ACPI_STATUS
204 AcpiExAccessRegion (
205     ACPI_OPERAND_OBJECT     *ObjDesc,
206     UINT32                  FieldDatumByteOffset,
207     UINT64                  *Value,
208     UINT32                  ReadWrite);


211 /*
212  * exmisc - misc support routines
213  */
214 ACPI_STATUS
215 AcpiExGetObjectReference (
216     ACPI_OPERAND_OBJECT     *ObjDesc,
217     ACPI_OPERAND_OBJECT     **ReturnDesc,
218     ACPI_WALK_STATE         *WalkState);

220 ACPI_STATUS
221 AcpiExConcatTemplate (
222     ACPI_OPERAND_OBJECT     *ObjDesc,
223     ACPI_OPERAND_OBJECT     *ObjDesc2,
224     ACPI_OPERAND_OBJECT     **ActualReturnDesc,
225     ACPI_WALK_STATE         *WalkState);

227 ACPI_STATUS
228 AcpiExDoConcatenate (
229     ACPI_OPERAND_OBJECT     *ObjDesc,
230     ACPI_OPERAND_OBJECT     *ObjDesc2,
231     ACPI_OPERAND_OBJECT     **ActualReturnDesc,
232     ACPI_WALK_STATE         *WalkState);

234 ACPI_STATUS
235 AcpiExDoLogicalNumericOp (
236     UINT16                  Opcode,
237     UINT64                  Integer0,
238     UINT64                  Integer1,
239     BOOLEAN                 *LogicalResult);

241 ACPI_STATUS
242 AcpiExDoLogicalOp (
243     UINT16                  Opcode,
244     ACPI_OPERAND_OBJECT     *Operand0,
245     ACPI_OPERAND_OBJECT     *Operand1,
246     BOOLEAN                 *LogicalResult);

248 UINT64
249 AcpiExDoMathOp (
250     UINT16                  Opcode,
251     UINT64                  Operand0,
252     UINT64                  Operand1);

254 ACPI_STATUS
255 AcpiExCreateMutex (
256     ACPI_WALK_STATE         *WalkState);
```

```
258 ACPI_STATUS
259 AcpiExCreateProcessor (
260     ACPI_WALK_STATE         *WalkState);

262 ACPI_STATUS
263 AcpiExCreatePowerResource (
264     ACPI_WALK_STATE         *WalkState);

266 ACPI_STATUS
267 AcpiExCreateRegion (
268     UINT8                   *AmlStart,
269     UINT32                  AmlLength,
270     UINT8                   RegionSpace,
271     ACPI_WALK_STATE         *WalkState);

273 ACPI_STATUS
274 AcpiExCreateEvent (
275     ACPI_WALK_STATE         *WalkState);

277 ACPI_STATUS
278 AcpiExCreateAlias (
279     ACPI_WALK_STATE         *WalkState);

281 ACPI_STATUS
282 AcpiExCreateMethod (
283     UINT8                   *AmlStart,
284     UINT32                  AmlLength,
285     ACPI_WALK_STATE         *WalkState);


288 /*
289  * exconfig - dynamic table load/unload
290  */
291 ACPI_STATUS
292 AcpiExLoadOp (
293     ACPI_OPERAND_OBJECT     *ObjDesc,
294     ACPI_OPERAND_OBJECT     *Target,
295     ACPI_WALK_STATE         *WalkState);

297 ACPI_STATUS
298 AcpiExLoadTableOp (
299     ACPI_WALK_STATE         *WalkState,
300     ACPI_OPERAND_OBJECT     **ReturnDesc);

302 ACPI_STATUS
303 AcpiExUnloadTable (
304     ACPI_OPERAND_OBJECT     *DdbHandle);


307 /*
308  * exmutex - mutex support
309  */
310 ACPI_STATUS
311 AcpiExAcquireMutex (
312     ACPI_OPERAND_OBJECT     *TimeDesc,
313     ACPI_OPERAND_OBJECT     *ObjDesc,
314     ACPI_WALK_STATE         *WalkState);

316 ACPI_STATUS
317 AcpiExAcquireMutexObject (
318     UINT16                  Timeout,
319     ACPI_OPERAND_OBJECT     *ObjDesc,
320     ACPI_THREAD_ID          ThreadId);

322 ACPI_STATUS
323 AcpiExReleaseMutex (
```



```
324     ACPI_OPERAND_OBJECT     *ObjDesc,
325     ACPI_WALK_STATE         *WalkState);

327 ACPI_STATUS
328 AcpiExReleaseMutexObject (
329     ACPI_OPERAND_OBJECT     *ObjDesc);

331 void
332 AcpiExReleaseAllMutexes (
333     ACPI_THREAD_STATE       *Thread);

335 void
336 AcpiExUnlinkMutex (
337     ACPI_OPERAND_OBJECT     *ObjDesc);


340 /*
341  * exprep - ACPI AML execution - prep utilities
342  */
343 ACPI_STATUS
344 AcpiExPrepCommonFieldObject (
345     ACPI_OPERAND_OBJECT     *ObjDesc,
346     UINT8                   FieldFlags,
347     UINT8                   FieldAttribute,
348     UINT32                  FieldBitPosition,
349     UINT32                  FieldBitLength);

351 ACPI_STATUS
352 AcpiExPrepFieldValue (
353     ACPI_CREATE_FIELD_INFO  *Info);


356 /*
357  * exsystem - Interface to OS services
358  */
359 ACPI_STATUS
360 AcpiExSystemDoNotifyOp (
361     ACPI_OPERAND_OBJECT     *Value,
362     ACPI_OPERAND_OBJECT     *ObjDesc);

364 ACPI_STATUS
365 AcpiExSystemDoSleep(
366     UINT64                  Time);

368 ACPI_STATUS
369 AcpiExSystemDoStall (
370     UINT32                  Time);

372 ACPI_STATUS
373 AcpiExSystemSignalEvent(
374     ACPI_OPERAND_OBJECT     *ObjDesc);

376 ACPI_STATUS
377 AcpiExSystemWaitEvent(
378     ACPI_OPERAND_OBJECT     *Time,
379     ACPI_OPERAND_OBJECT     *ObjDesc);

381 ACPI_STATUS
382 AcpiExSystemResetEvent(
383     ACPI_OPERAND_OBJECT     *ObjDesc);

385 ACPI_STATUS
386 AcpiExSystemWaitSemaphore (
387     ACPI_SEMAPHORE          Semaphore,
388     UINT16                  Timeout);
```

```
390 ACPI_STATUS
391 AcpiExSystemWaitMutex (
392     ACPI_MUTEX              Mutex,
393     UINT16                  Timeout);

395 /*
396  * exoparg1 - ACPI AML execution, 1 operand
397  */
398 ACPI_STATUS
399 AcpiExOpcode_0A_0T_1R (
400     ACPI_WALK_STATE         *WalkState);

402 ACPI_STATUS
403 AcpiExOpcode_1A_0T_0R (
404     ACPI_WALK_STATE         *WalkState);

406 ACPI_STATUS
407 AcpiExOpcode_1A_0T_1R (
408     ACPI_WALK_STATE         *WalkState);

410 ACPI_STATUS
411 AcpiExOpcode_1A_1T_1R (
412     ACPI_WALK_STATE         *WalkState);

414 ACPI_STATUS
415 AcpiExOpcode_1A_1T_0R (
416     ACPI_WALK_STATE         *WalkState);

418 /*
419  * exoparg2 - ACPI AML execution, 2 operands
420  */
421 ACPI_STATUS
422 AcpiExOpcode_2A_0T_0R (
423     ACPI_WALK_STATE         *WalkState);

425 ACPI_STATUS
426 AcpiExOpcode_2A_0T_1R (
427     ACPI_WALK_STATE         *WalkState);

429 ACPI_STATUS
430 AcpiExOpcode_2A_1T_1R (
431     ACPI_WALK_STATE         *WalkState);

433 ACPI_STATUS
434 AcpiExOpcode_2A_2T_1R (
435     ACPI_WALK_STATE         *WalkState);


438 /*
439  * exoparg3 - ACPI AML execution, 3 operands
440  */
441 ACPI_STATUS
442 AcpiExOpcode_3A_0T_0R (
443     ACPI_WALK_STATE         *WalkState);

445 ACPI_STATUS
446 AcpiExOpcode_3A_1T_1R (
447     ACPI_WALK_STATE         *WalkState);


450 /*
451  * exoparg6 - ACPI AML execution, 6 operands
452  */
453 ACPI_STATUS
454 AcpiExOpcode_6A_0T_1R (
455     ACPI_WALK_STATE         *WalkState);
```



```
458 /*
459  * exresolv - Object resolution and get value functions
460  */
461 ACPI_STATUS
462 AcpiExResolveToValue (
463     ACPI_OPERAND_OBJECT     **StackPtr,
464     ACPI_WALK_STATE         *WalkState);

466 ACPI_STATUS
467 AcpiExResolveMultiple (
468     ACPI_WALK_STATE         *WalkState,
469     ACPI_OPERAND_OBJECT     *Operand,
470     ACPI_OBJECT_TYPE        *ReturnType,
471     ACPI_OPERAND_OBJECT     **ReturnDesc);


474 /*
475  * exresnte - resolve namespace node
476  */
477 ACPI_STATUS
478 AcpiExResolveNodeToValue (
479     ACPI_NAMESPACE_NODE     **StackPtr,
480     ACPI_WALK_STATE         *WalkState);


483 /*
484  * exresop - resolve operand to value
485  */
486 ACPI_STATUS
487 AcpiExResolveOperands (
488     UINT16                  Opcode,
489     ACPI_OPERAND_OBJECT     **StackPtr,
490     ACPI_WALK_STATE         *WalkState);


493 /*
494  * exdump - Interpreter debug output routines
495  */
496 void
497 AcpiExDumpOperand (
498     ACPI_OPERAND_OBJECT     *ObjDesc,
499     UINT32                  Depth);

501 void
502 AcpiExDumpOperands (
503     ACPI_OPERAND_OBJECT     **Operands,
504     const char              *OpcodeName,
505     UINT32                  NumOpcodes);

507 void
508 AcpiExDumpObjectDescriptor (
509     ACPI_OPERAND_OBJECT     *Object,
510     UINT32                  Flags);

512 void
513 AcpiExDumpNamespaceNode (
514     ACPI_NAMESPACE_NODE     *Node,
515     UINT32                  Flags);


518 /*
519  * exnames - AML namestring support
520  */
521 ACPI_STATUS
```

```
AcpiExGetNameString (
    ACPI_OBJECT_TYPE        DataType,
    UINT8                   *InAmlAddress,
    char                    **OutNameString,
    UINT32                  *OutNameLength);


/*
 * exstore - Object store support
 */
ACPI_STATUS
AcpiExStore (
    ACPI_OPERAND_OBJECT     *ValDesc,
    ACPI_OPERAND_OBJECT     *DestDesc,
    ACPI_WALK_STATE         *WalkState);

ACPI_STATUS
AcpiExStoreObjectToNode (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_WALK_STATE         *WalkState,
    UINT8                   ImplicitConversion);

#define ACPI_IMPLICIT_CONVERSION        TRUE
#define ACPI_NO_IMPLICIT_CONVERSION     FALSE


/*
 * exstoren - resolve/store object
 */
ACPI_STATUS
AcpiExResolveObject (
    ACPI_OPERAND_OBJECT     **SourceDescPtr,
    ACPI_OBJECT_TYPE        TargetType,
    ACPI_WALK_STATE         *WalkState);

ACPI_STATUS
AcpiExStoreObjectToObject (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *DestDesc,
    ACPI_OPERAND_OBJECT     **NewDesc,
    ACPI_WALK_STATE         *WalkState);


/*
 * exstorob - store object - buffer/string
 */
ACPI_STATUS
AcpiExStoreBufferToBuffer (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc);

ACPI_STATUS
AcpiExStoreStringToString (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc);


/*
 * excopy - object copy
 */
ACPI_STATUS
AcpiExCopyIntegerToIndexField (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc);
```



```
ACPI_STATUS
AcpiExCopyIntegerToBankField (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc);

ACPI_STATUS
AcpiExCopyDataToNamedField (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_NAMESPACE_NODE     *Node);

ACPI_STATUS
AcpiExCopyIntegerToBufferField (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *TargetDesc);


/*
 * exutils - interpreter/scanner utilities
 */
void
AcpiExEnterInterpreter (
    void);

void
AcpiExExitInterpreter (
    void);

void
AcpiExReacquireInterpreter (
    void);

void
AcpiExRelinquishInterpreter (
    void);

BOOLEAN
void
AcpiExTruncateFor32bitTable (
    ACPI_OPERAND_OBJECT     *ObjDesc);

void
AcpiExAcquireGlobalLock (
    UINT32                  Rule);

void
AcpiExReleaseGlobalLock (
    UINT32                  Rule);

void
AcpiExEisaIdToString (
    char                    *Dest,
    UINT64                  CompressedId);

void
AcpiExIntegerToString (
    char                    *Dest,
    UINT64                  Value);

BOOLEAN
AcpiIsValidSpaceId (
    UINT8                   SpaceId);


/*
 * exregion - default OpRegion handlers
 */
```

```
ACPI_STATUS
AcpiExSystemMemorySpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExSystemIoSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExPciConfigSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExCmosSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExPciBarSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExEmbeddedControllerSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

ACPI_STATUS
AcpiExSmBusSpaceHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);


ACPI_STATUS
AcpiExDataTableSpaceHandler (
```



```
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext);

#endif /* __INTERP_H__ */
```

```
**********************************************************
   47041 Thu Dec 26 13:49:49 2013
new/usr/src/common/acpica/include/aclocal.h
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: aclocal.h - Internal data types used across the ACPI subsystem
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACLOCAL_H__
#define __ACLOCAL_H__


/* acpisrc:StructDefs -- for acpisrc conversion */

#define ACPI_SERIALIZED                 0xFF

typedef UINT32                          ACPI_MUTEX_HANDLE;
#define ACPI_GLOBAL_LOCK                (ACPI_SEMAPHORE) (-1)

/* Total number of aml opcodes defined */

#define AML_NUM_OPCODES                 0x81
```



```
#define AML_NUM_OPCODES                 0x7F


/* Forward declarations */

struct acpi_walk_state;
struct acpi_obj_mutex;
union acpi_parse_object;


/*****************************************************************************
 *
 * Mutex typedefs and structs
 *
 ****************************************************************************/


/*
 * Predefined handles for the mutex objects used within the subsystem
 * All mutex objects are automatically created by AcpiUtMutexInitialize.
 *
 * The acquire/release ordering protocol is implied via this list. Mutexes
 * with a lower value must be acquired before mutexes with a higher value.
 *
 * NOTE: any changes here must be reflected in the AcpiGbl_MutexNames
 * table below also!
 */
#define ACPI_MTX_INTERPRETER            0   /* AML Interpreter, main lock */
#define ACPI_MTX_NAMESPACE              1   /* ACPI Namespace */
#define ACPI_MTX_TABLES                 2   /* Data for ACPI tables */
#define ACPI_MTX_EVENTS                 3   /* Data for ACPI events */
#define ACPI_MTX_CACHES                 4   /* Internal caches, general purposes
#define ACPI_MTX_MEMORY                 5   /* Debug memory tracking lists */
#define ACPI_MTX_DEBUG_CMD_COMPLETE     6   /* AML debugger */
#define ACPI_MTX_DEBUG_CMD_READY        7   /* AML debugger */

#define ACPI_MAX_MUTEX                  7
#define ACPI_NUM_MUTEX                  ACPI_MAX_MUTEX+1


/* Lock structure for reader/writer interfaces */

typedef struct acpi_rw_lock
{
    ACPI_MUTEX              WriterMutex;
    ACPI_MUTEX              ReaderMutex;
    UINT32                  NumReaders;

} ACPI_RW_LOCK;
______unchanged_portion_omitted_


/* Namespace Node flags */

#define ANOBJ_RESERVED                  0x01    /* Available for use */
#define ANOBJ_TEMPORARY                 0x02    /* Node is create by a method an
#define ANOBJ_METHOD_ARG                0x04    /* Node is a method argument */
#define ANOBJ_METHOD_LOCAL              0x08    /* Node is a method local */
#define ANOBJ_SUBTREE_HAS_INI           0x10    /* Used to optimize device initi
#define ANOBJ_EVALUATED                 0x20    /* Set on first evaluation of no
#define ANOBJ_ALLOCATED_BUFFER          0x40    /* Method AML buffer is dynamic

#define ANOBJ_IS_EXTERNAL               0x08    /* iASL only: This object create
#define ANOBJ_METHOD_NO_RETVAL          0x10    /* iASL only: Method has no retu
#define ANOBJ_METHOD_SOME_NO_RETVAL     0x20    /* iASL only: Method has at leas
#define ANOBJ_IS_BIT_OFFSET             0x40    /* iASL only: Reference is a bit
```

```
216 #define ANOBJ_IS_REFERENCED             0x80    /* iASL only: Object was referen


219 /* Internal ACPI table management - master table list */

221 typedef struct acpi_table_list
222 {
223     ACPI_TABLE_DESC                 *Tables;            /* Table descriptor arra
224     UINT32                          CurrentTableCount;  /* Tables currently in t
225     UINT32                          MaxTableCount;      /* Max tables array will
226     UINT8                           Flags;

228 } ACPI_TABLE_LIST;
```

*unchanged_portion_omitted_*

```
280 /* Field creation info */

282 typedef struct acpi_create_field_info
283 {
284     ACPI_NAMESPACE_NODE             *RegionNode;
285     ACPI_NAMESPACE_NODE             *FieldNode;
286     ACPI_NAMESPACE_NODE             *RegisterNode;
287     ACPI_NAMESPACE_NODE             *DataRegisterNode;
288     ACPI_NAMESPACE_NODE             *ConnectionNode;
289     UINT8                           *ResourceBuffer;
290     UINT32                          BankValue;
291     UINT32                          FieldBitPosition;
292     UINT32                          FieldBitLength;
293     UINT16                          ResourceLength;
294     UINT8                           FieldFlags;
295     UINT8                           Attribute;
296     UINT8                           FieldType;
297     UINT8                           AccessLength;

299 } ACPI_CREATE_FIELD_INFO;


302 typedef
303 ACPI_STATUS (*ACPI_INTERNAL_METHOD) (
304     struct acpi_walk_state          *WalkState);


307 /*
308  * Bitmapped ACPI types. Used internally only
309  */
310 #define ACPI_BTYPE_ANY                  0x00000000
311 #define ACPI_BTYPE_INTEGER              0x00000001
312 #define ACPI_BTYPE_STRING               0x00000002
313 #define ACPI_BTYPE_BUFFER               0x00000004
314 #define ACPI_BTYPE_PACKAGE              0x00000008
315 #define ACPI_BTYPE_FIELD_UNIT           0x00000010
316 #define ACPI_BTYPE_DEVICE               0x00000020
317 #define ACPI_BTYPE_EVENT                0x00000040
318 #define ACPI_BTYPE_METHOD               0x00000080
319 #define ACPI_BTYPE_MUTEX                0x00000100
320 #define ACPI_BTYPE_REGION               0x00000200
321 #define ACPI_BTYPE_POWER                0x00000400
322 #define ACPI_BTYPE_PROCESSOR            0x00000800
323 #define ACPI_BTYPE_THERMAL              0x00001000
324 #define ACPI_BTYPE_BUFFER_FIELD         0x00002000
325 #define ACPI_BTYPE_DDB_HANDLE           0x00004000
326 #define ACPI_BTYPE_DEBUG_OBJECT         0x00008000
327 #define ACPI_BTYPE_REFERENCE            0x00010000
328 #define ACPI_BTYPE_RESOURCE             0x00020000
```



```
330 #define ACPI_BTYPE_COMPUTE_DATA         (ACPI_BTYPE_INTEGER | ACPI_BTYPE_STRING

332 #define ACPI_BTYPE_DATA                 (ACPI_BTYPE_COMPUTE_DATA  | ACPI_BTYPE_P
333 #define ACPI_BTYPE_DATA_REFERENCE       (ACPI_BTYPE_DATA | ACPI_BTYPE_REFERENCE
334 #define ACPI_BTYPE_DEVICE_OBJECTS       (ACPI_BTYPE_DEVICE | ACPI_BTYPE_THERMAL
335 #define ACPI_BTYPE_OBJECTS_AND_REFS     0x0001FFFF  /* ARG or LOCAL */
336 #define ACPI_BTYPE_ALL_OBJECTS          0x0000FFFF

338 #pragma pack(1)

340 /*
341  * Information structure for ACPI predefined names.
342  * Each entry in the table contains the following items:
343  *
344  * Name                 - The ACPI reserved name
345  * ParamCount           - Number of arguments to the method
346  * ExpectedReturnBtypes - Allowed type(s) for the return value
347  */
348 typedef struct acpi_name_info
349 {
350     char                            Name[ACPI_NAME_SIZE];
351     UINT16                          ArgumentList;
347     UINT8                           ParamCount;
352     UINT8                           ExpectedBtypes;

354 } ACPI_NAME_INFO;

356 /*
357  * Secondary information structures for ACPI predefined objects that return
358  * package objects. This structure appears as the next entry in the table
359  * after the NAME_INFO structure above.
360  *
361  * The reason for this is to minimize the size of the predefined name table.
362  */

364 /*
365  * Used for ACPI_PTYPE1_FIXED, ACPI_PTYPE1_VAR, ACPI_PTYPE2,
366  * ACPI_PTYPE2_MIN, ACPI_PTYPE2_PKG_COUNT, ACPI_PTYPE2_COUNT,
367  * ACPI_PTYPE2_FIX_VAR
362  * ACPI_PTYPE2_MIN, ACPI_PTYPE2_PKG_COUNT, ACPI_PTYPE2_COUNT
368  */
369 typedef struct acpi_package_info
370 {
371     UINT8                           Type;
372     UINT8                           ObjectType1;
373     UINT8                           Count1;
374     UINT8                           ObjectType2;
375     UINT8                           Count2;
376     UINT16                          Reserved;
371     UINT8                           Reserved;

378 } ACPI_PACKAGE_INFO;

380 /* Used for ACPI_PTYPE2_FIXED */

382 typedef struct acpi_package_info2
383 {
384     UINT8                           Type;
385     UINT8                           Count;
386     UINT8                           ObjectType[4];
387     UINT8                           Reserved;

389 } ACPI_PACKAGE_INFO2;

391 /* Used for ACPI_PTYPE1_OPTION */
```

```
typedef struct acpi_package_info3
{
    UINT8                       Type;
    UINT8                       Count;
    UINT8                       ObjectType[2];
    UINT8                       TailObjectType;
    UINT16                      Reserved;
    UINT8                       Reserved;

} ACPI_PACKAGE_INFO3;
```

*unchanged_portion_omitted_*

```
/* Reset to default packing */

#pragma pack()
/* Data block used during object validation */

typedef struct acpi_predefined_data
{
    char                        *Pathname;
    const ACPI_PREDEFINED_INFO  *Predefined;
    union acpi_operand_object   *ParentPackage;
    UINT32                      Flags;
    UINT8                       NodeFlags;

/* Return object auto-repair info */
} ACPI_PREDEFINED_DATA;

typedef ACPI_STATUS (*ACPI_OBJECT_CONVERTER) (
    union acpi_operand_object   *OriginalObject,
    union acpi_operand_object   **ConvertedObject);
/* Defines for Flags field above */

typedef struct acpi_simple_repair_info
{
    char                        Name[ACPI_NAME_SIZE];
    UINT32                      UnexpectedBtypes;
    UINT32                      PackageIndex;
    ACPI_OBJECT_CONVERTER       ObjectConverter;
#define ACPI_OBJECT_REPAIRED    1

} ACPI_SIMPLE_REPAIR_INFO;


/*
 * Bitmapped return value types
 * Note: the actual data types must be contiguous, a loop in nspredef.c
 * depends on this.
 */
#define ACPI_RTYPE_ANY                  0x00
#define ACPI_RTYPE_NONE                 0x01
#define ACPI_RTYPE_INTEGER              0x02
#define ACPI_RTYPE_STRING               0x04
#define ACPI_RTYPE_BUFFER               0x08
#define ACPI_RTYPE_PACKAGE              0x10
#define ACPI_RTYPE_REFERENCE            0x20
#define ACPI_RTYPE_ALL                  0x3F

#define ACPI_NUM_RTYPES                 5   /* Number of actual object types */


/*****************************************************************************
 *
 * Event typedefs and structs
 *
 ****************************************************************************/
```



```
/* Dispatch info for each host-installed SCI handler */

typedef struct acpi_sci_handler_info
{
    struct acpi_sci_handler_info    *Next;
    ACPI_SCI_HANDLER                Address;        /* Address of handler */
    void                            *Context;       /* Context to be passed to h

} ACPI_SCI_HANDLER_INFO;

/* Dispatch info for each GPE -- either a method or handler, cannot be both */

typedef struct acpi_gpe_handler_info
{
    ACPI_GPE_HANDLER                Address;        /* Address of handler, if an
    void                            *Context;       /* Context to be passed to h
    ACPI_NAMESPACE_NODE             *MethodNode;    /* Method node for this GPE
    UINT8                           OriginalFlags;  /* Original (pre-handler) GP
    BOOLEAN                         OriginallyEnabled; /* True if GPE was origin

} ACPI_GPE_HANDLER_INFO;

/* Notify info for implicit notify, multiple device objects */

typedef struct acpi_gpe_notify_info
{
    ACPI_NAMESPACE_NODE             *DeviceNode;    /* Device to be notified */
    struct acpi_gpe_notify_info     *Next;

} ACPI_GPE_NOTIFY_INFO;

/*
 * GPE dispatch info. At any time, the GPE can have at most one type
 * of dispatch - Method, Handler, or Implicit Notify.
 */
typedef union acpi_gpe_dispatch_info
{
    ACPI_NAMESPACE_NODE             *MethodNode;    /* Method node for this GPE
    ACPI_GPE_HANDLER_INFO           *Handler;       /* Installed GPE handler */
    ACPI_GPE_NOTIFY_INFO            *NotifyList;    /* List of _PRW devices for
    struct acpi_gpe_handler_info    *Handler;       /* Installed GPE handler */
    ACPI_NAMESPACE_NODE             *DeviceNode;    /* Parent _PRW device for im

} ACPI_GPE_DISPATCH_INFO;

/*
 * Information about a GPE, one per each GPE in an array.
 * NOTE: Important to keep this struct as small as possible.
 */
typedef struct acpi_gpe_event_info
{
    union acpi_gpe_dispatch_info    Dispatch;       /* Either Method, Handler, o
    union acpi_gpe_dispatch_info    Dispatch;       /* Either Method or Handler
    struct acpi_gpe_register_info   *RegisterInfo;  /* Backpointer to register i
    UINT8                           Flags;          /* Misc info about this GPE
    UINT8                           GpeNumber;      /* This GPE */
    UINT8                           RuntimeCount;   /* References to a run GPE *

} ACPI_GPE_EVENT_INFO;
```

*unchanged_portion_omitted_*

```
typedef
ACPI_STATUS (*ACPI_PARSE_DOWNWARDS) (
    struct acpi_walk_state          *WalkState,
```

```
    union acpi_parse_object         **OutOp);

typedef
ACPI_STATUS (*ACPI_PARSE_UPWARDS) (
    struct acpi_walk_state          *WalkState);


/* Global handlers for AML Notifies */

typedef struct acpi_global_notify_handler
{
    ACPI_NOTIFY_HANDLER             Handler;
    void                            *Context;

} ACPI_GLOBAL_NOTIFY_HANDLER;

/*
 * Notify info - used to pass info to the deferred notify
 * handler/dispatcher.
 */
typedef struct acpi_notify_info
{
    ACPI_STATE_COMMON
    UINT8                           HandlerListId;
    ACPI_NAMESPACE_NODE             *Node;
    union acpi_operand_object       *HandlerListHead;
    ACPI_GLOBAL_NOTIFY_HANDLER      *Global;
    union acpi_operand_object       *HandlerObj;

} ACPI_NOTIFY_INFO;
```

*unchanged_portion_omitted_*

```
/*****************************************************************************
 *
 * Interpreter typedefs and structs
 *
 ****************************************************************************/

typedef
ACPI_STATUS (*ACPI_EXECUTE_OP) (
    struct acpi_walk_state          *WalkState);

/* Address Range info block */

typedef struct acpi_address_range
{
    struct acpi_address_range   *Next;
    ACPI_NAMESPACE_NODE         *RegionNode;
    ACPI_PHYSICAL_ADDRESS       StartAddress;
    ACPI_PHYSICAL_ADDRESS       EndAddress;

} ACPI_ADDRESS_RANGE;


/*****************************************************************************
 *
 * Parser typedefs and structs
 *
 ****************************************************************************/

/*
 * AML opcode, name, and argument layout
 */
typedef struct acpi_opcode_info
{
```



```
#if defined(ACPI_DISASSEMBLER) || defined(ACPI_DEBUG_OUTPUT)
    char                            *Name;          /* Opcode name (disassembler
#endif
    UINT32                          ParseArgs;      /* Grammar/Parse time argume
    UINT32                          RuntimeArgs;    /* Interpret time arguments
    UINT16                          Flags;          /* Misc flags */
    UINT8                           ObjectType;     /* Corresponding internal ob
    UINT8                           Class;          /* Opcode class */
    UINT8                           Type;           /* Opcode type */

} ACPI_OPCODE_INFO;

/* Structure for Resource Tag information */

typedef struct acpi_tag_info
{
    UINT32                          BitOffset;
    UINT32                          BitLength;

} ACPI_TAG_INFO;

/* Value associated with the parse object */

typedef union acpi_parse_value
{
    UINT64                          Integer;        /* Integer constant (Up to 6
    UINT32                          Size;           /* bytelist or field size */
    char                            *String;        /* NULL terminated string */
    UINT8                           *Buffer;        /* buffer or string */
    char                            *Name;          /* NULL terminated string */
    union acpi_parse_object         *Arg;           /* arguments and contained o
    ACPI_TAG_INFO                   Tag;            /* Resource descriptor tag i

} ACPI_PARSE_VALUE;


#ifdef ACPI_DISASSEMBLER
#define ACPI_DISASM_ONLY_MEMBERS(a)     a;
#else
#define ACPI_DISASM_ONLY_MEMBERS(a)
#endif

#define ACPI_PARSE_COMMON \
    union acpi_parse_object         *Parent;        /* Parent op */\
    UINT8                           DescriptorType; /* To differentiate various
    UINT8                           Flags;          /* Type of Op */\
    UINT16                          AmlOpcode;      /* AML opcode */\
    UINT32                          AmlOffset;      /* Offset of declaration in
    union acpi_parse_object         *Next;          /* Next op */\
    ACPI_NAMESPACE_NODE             *Node;          /* For use by interpreter */
    ACPI_PARSE_VALUE                Value;          /* Value or args associated
    UINT8                           ArgListLength;  /* Number of elements in the
    ACPI_DISASM_ONLY_MEMBERS (\
    UINT8                           DisasmFlags;    /* Used during AML disassemb
    UINT8                           DisasmOpcode;   /* Subtype used for disassem
    char                            AmlOpName[16])  /* Op name (debug only) */


/* Flags for DisasmFlags field  above */
#define ACPI_DASM_BUFFER                0x00
#define ACPI_DASM_RESOURCE              0x01
#define ACPI_DASM_STRING                0x02
#define ACPI_DASM_UNICODE               0x03
#define ACPI_DASM_EISAID                0x04
#define ACPI_DASM_MATCHOP               0x05
#define ACPI_DASM_LNOT_PREFIX           0x06
```

```
 818 #define ACPI_DASM_LNOT_SUFFIX           0x07
 819 #define ACPI_DASM_IGNORE                0x08

 875 #define ACPI_DASM_BUFFER                0x00        /* Buffer is a simple data b
 876 #define ACPI_DASM_RESOURCE              0x01        /* Buffer is a Resource Desc
 877 #define ACPI_DASM_STRING                0x02        /* Buffer is a ASCII string
 878 #define ACPI_DASM_UNICODE               0x03        /* Buffer is a Unicode strin
 879 #define ACPI_DASM_PLD_METHOD            0x04        /* Buffer is a _PLD method b
 880 #define ACPI_DASM_EISAID                0x05        /* Integer is an EISAID */
 881 #define ACPI_DASM_MATCHOP               0x06        /* Parent opcode is a Match(
 882 #define ACPI_DASM_LNOT_PREFIX           0x07        /* Start of a LNotEqual (etc
 883 #define ACPI_DASM_LNOT_SUFFIX           0x08        /* End  of a LNotEqual (etc.
 884 #define ACPI_DASM_IGNORE                0x09        /* Not used at this time */

 886 /*
 887  * Generic operation (for example:  If, While, Store)
 888  */
 889 typedef struct acpi_parse_obj_common
 890 {
 891     ACPI_PARSE_COMMON
 892 } ACPI_PARSE_OBJ_COMMON;
_____unchanged_portion_omitted_


 973 /* Parse object flags */

 975 #define ACPI_PARSEOP_GENERIC            0x01
 976 #define ACPI_PARSEOP_NAMED              0x02
 977 #define ACPI_PARSEOP_DEFERRED           0x04
 978 #define ACPI_PARSEOP_BYTELIST           0x08
 979 #define ACPI_PARSEOP_IN_STACK           0x10
 980 #define ACPI_PARSEOP_TARGET             0x20
 981 #define ACPI_PARSEOP_IN_CACHE           0x80

 983 /* Parse object DisasmFlags */

 985 #define ACPI_PARSEOP_IGNORE             0x01
 986 #define ACPI_PARSEOP_PARAMLIST          0x02
 987 #define ACPI_PARSEOP_EMPTY_TERMLIST     0x04
 988 #define ACPI_PARSEOP_PREDEF_CHECKED     0x08
 989 #define ACPI_PARSEOP_SPECIAL            0x10


 992 /*****************************************************************************
 993  *
 994  * Hardware (ACPI registers) and PNP
 995  *
 996  ****************************************************************************/

 998 typedef struct acpi_bit_register_info
 999 {
1000     UINT8                           ParentRegister;
1001     UINT8                           BitPosition;
1002     UINT16                          AccessBitMask;

1004 } ACPI_BIT_REGISTER_INFO;


1007 /*
1008  * Some ACPI registers have bits that must be ignored -- meaning that they
1009  * must be preserved.
1010  */
1011 #define ACPI_PM1_STATUS_PRESERVED_BITS          0x0800  /* Bit 11 */

1013 /* Write-only bits must be zeroed by software */
```



```
1015 #define ACPI_PM1_CONTROL_WRITEONLY_BITS         0x2004  /* Bits 13, 2 */

1017 /* For control registers, both ignored and reserved bits must be preserved */

1019 /*
1020  * For PM1 control, the SCI enable bit (bit 0, SCI_EN) is defined by the
1021  * ACPI specification to be a "preserved" bit - "OSPM always preserves this
1022  * bit position", section 4.7.3.2.1. However, on some machines the OS must
1023  * write a one to this bit after resume for the machine to work properly.
1024  * To enable this, we no longer attempt to preserve this bit. No machines
1025  * are known to fail if the bit is not preserved. (May 2009)
1026  */
1027 #define ACPI_PM1_CONTROL_IGNORED_BITS           0x0200  /* Bit 9 */
1028 #define ACPI_PM1_CONTROL_RESERVED_BITS          0xC1F8  /* Bits 14-15, 3-8 */
1029 #define ACPI_PM1_CONTROL_PRESERVED_BITS \
1030          (ACPI_PM1_CONTROL_IGNORED_BITS | ACPI_PM1_CONTROL_RESERVED_BITS)

1032 #define ACPI_PM2_CONTROL_PRESERVED_BITS         0xFFFFFFFE /* All except bit 0 *

1034 /*
1035  * Register IDs
1036  * These are the full ACPI registers
1037  */
1038 #define ACPI_REGISTER_PM1_STATUS                0x01
1039 #define ACPI_REGISTER_PM1_ENABLE                0x02
1040 #define ACPI_REGISTER_PM1_CONTROL               0x03
1041 #define ACPI_REGISTER_PM2_CONTROL               0x04
1042 #define ACPI_REGISTER_PM_TIMER                  0x05
1043 #define ACPI_REGISTER_PROCESSOR_BLOCK           0x06
1044 #define ACPI_REGISTER_SMI_COMMAND_BLOCK         0x07


1047 /* Masks used to access the BitRegisters */

1049 #define ACPI_BITMASK_TIMER_STATUS               0x0001
1050 #define ACPI_BITMASK_BUS_MASTER_STATUS          0x0010
1051 #define ACPI_BITMASK_GLOBAL_LOCK_STATUS         0x0020
1052 #define ACPI_BITMASK_POWER_BUTTON_STATUS        0x0100
1053 #define ACPI_BITMASK_SLEEP_BUTTON_STATUS        0x0200
1054 #define ACPI_BITMASK_RT_CLOCK_STATUS            0x0400
1055 #define ACPI_BITMASK_PCIEXP_WAKE_STATUS         0x4000    /* ACPI 3.0 */
1056 #define ACPI_BITMASK_WAKE_STATUS                0x8000

1058 #define ACPI_BITMASK_ALL_FIXED_STATUS           (\
1059     ACPI_BITMASK_TIMER_STATUS          | \
1060     ACPI_BITMASK_BUS_MASTER_STATUS     | \
1061     ACPI_BITMASK_GLOBAL_LOCK_STATUS    | \
1062     ACPI_BITMASK_POWER_BUTTON_STATUS   | \
1063     ACPI_BITMASK_SLEEP_BUTTON_STATUS   | \
1064     ACPI_BITMASK_RT_CLOCK_STATUS       | \
1065     ACPI_BITMASK_PCIEXP_WAKE_STATUS    | \
1066     ACPI_BITMASK_WAKE_STATUS)

1068 #define ACPI_BITMASK_TIMER_ENABLE               0x0001
1069 #define ACPI_BITMASK_GLOBAL_LOCK_ENABLE         0x0020
1070 #define ACPI_BITMASK_POWER_BUTTON_ENABLE        0x0100
1071 #define ACPI_BITMASK_SLEEP_BUTTON_ENABLE        0x0200
1072 #define ACPI_BITMASK_RT_CLOCK_ENABLE            0x0400
1073 #define ACPI_BITMASK_PCIEXP_WAKE_DISABLE        0x4000    /* ACPI 3.0 */

1075 #define ACPI_BITMASK_SCI_ENABLE                 0x0001
1076 #define ACPI_BITMASK_BUS_MASTER_RLD             0x0002
1077 #define ACPI_BITMASK_GLOBAL_LOCK_RELEASE        0x0004
1078 #define ACPI_BITMASK_SLEEP_TYPE                 0x1C00
1079 #define ACPI_BITMASK_SLEEP_ENABLE               0x2000
```

```
1081 #define ACPI_BITMASK_ARB_DISABLE                0x0001


1084 /* Raw bit position of each BitRegister */

1086 #define ACPI_BITPOSITION_TIMER_STATUS           0x00
1087 #define ACPI_BITPOSITION_BUS_MASTER_STATUS      0x04
1088 #define ACPI_BITPOSITION_GLOBAL_LOCK_STATUS     0x05
1089 #define ACPI_BITPOSITION_POWER_BUTTON_STATUS    0x08
1090 #define ACPI_BITPOSITION_SLEEP_BUTTON_STATUS    0x09
1091 #define ACPI_BITPOSITION_RT_CLOCK_STATUS        0x0A
1092 #define ACPI_BITPOSITION_PCIEXP_WAKE_STATUS     0x0E    /* ACPI 3.0 */
1093 #define ACPI_BITPOSITION_WAKE_STATUS            0x0F

1095 #define ACPI_BITPOSITION_TIMER_ENABLE           0x00
1096 #define ACPI_BITPOSITION_GLOBAL_LOCK_ENABLE     0x05
1097 #define ACPI_BITPOSITION_POWER_BUTTON_ENABLE    0x08
1098 #define ACPI_BITPOSITION_SLEEP_BUTTON_ENABLE    0x09
1099 #define ACPI_BITPOSITION_RT_CLOCK_ENABLE        0x0A
1100 #define ACPI_BITPOSITION_PCIEXP_WAKE_DISABLE    0x0E    /* ACPI 3.0 */

1102 #define ACPI_BITPOSITION_SCI_ENABLE             0x00
1103 #define ACPI_BITPOSITION_BUS_MASTER_RLD         0x01
1104 #define ACPI_BITPOSITION_GLOBAL_LOCK_RELEASE    0x02
1105 #define ACPI_BITPOSITION_SLEEP_TYPE             0x0A
1106 #define ACPI_BITPOSITION_SLEEP_ENABLE           0x0D

1108 #define ACPI_BITPOSITION_ARB_DISABLE            0x00


1111 /* Structs and definitions for _OSI support and I/O port validation */

1047 #define ACPI_OSI_WIN_2000               0x01
1048 #define ACPI_OSI_WIN_XP                 0x02
1049 #define ACPI_OSI_WIN_XP_SP1             0x03
1050 #define ACPI_OSI_WINSRV_2003            0x04
1051 #define ACPI_OSI_WIN_XP_SP2             0x05
1052 #define ACPI_OSI_WINSRV_2003_SP1        0x06
1053 #define ACPI_OSI_WIN_VISTA              0x07
1054 #define ACPI_OSI_WINSRV_2008            0x08
1055 #define ACPI_OSI_WIN_VISTA_SP1          0x09
1056 #define ACPI_OSI_WIN_VISTA_SP2          0x0A
1057 #define ACPI_OSI_WIN_7                  0x0B

1113 #define ACPI_ALWAYS_ILLEGAL             0x00

1115 typedef struct acpi_interface_info
1116 {
1117     char                        *Name;
1118     struct acpi_interface_info  *Next;
1119     UINT8                       Flags;
1120     UINT8                       Value;

1122 } ACPI_INTERFACE_INFO;

1124 #define ACPI_OSI_INVALID                0x01
1125 #define ACPI_OSI_DYNAMIC                0x02
1126 #define ACPI_OSI_FEATURE                0x04
1127 #define ACPI_OSI_DEFAULT_INVALID        0x08
1128 #define ACPI_OSI_OPTIONAL_FEATURE       (ACPI_OSI_FEATURE | ACPI_OSI_DEFAULT_INV

1130 typedef struct acpi_port_info
1131 {
1132     char                    *Name;
1133     UINT16                  Start;
1134     UINT16                  End;
```



```
1135     UINT8                   OsiDependency;

1137 } ACPI_PORT_INFO;


1140 /*****************************************************************************
1141  *
1142  * Resource descriptors
1143  *
1144  ****************************************************************************/

1146 /* ResourceType values */

1148 #define ACPI_ADDRESS_TYPE_MEMORY_RANGE          0
1149 #define ACPI_ADDRESS_TYPE_IO_RANGE              1
1150 #define ACPI_ADDRESS_TYPE_BUS_NUMBER_RANGE      2

1152 /* Resource descriptor types and masks */

1154 #define ACPI_RESOURCE_NAME_LARGE                0x80
1155 #define ACPI_RESOURCE_NAME_SMALL                0x00

1157 #define ACPI_RESOURCE_NAME_SMALL_MASK           0x78 /* Bits 6:3 contain the typ
1158 #define ACPI_RESOURCE_NAME_SMALL_LENGTH_MASK    0x07 /* Bits 2:0 contain the len
1159 #define ACPI_RESOURCE_NAME_LARGE_MASK           0x7F /* Bits 6:0 contain the typ


1162 /*
1163  * Small resource descriptor "names" as defined by the ACPI specification.
1164  * Note: Bits 2:0 are used for the descriptor length
1165  */
1166 #define ACPI_RESOURCE_NAME_IRQ                  0x20
1167 #define ACPI_RESOURCE_NAME_DMA                  0x28
1168 #define ACPI_RESOURCE_NAME_START_DEPENDENT      0x30
1169 #define ACPI_RESOURCE_NAME_END_DEPENDENT        0x38
1170 #define ACPI_RESOURCE_NAME_IO                   0x40
1171 #define ACPI_RESOURCE_NAME_FIXED_IO             0x48
1172 #define ACPI_RESOURCE_NAME_FIXED_DMA            0x50
1115 #define ACPI_RESOURCE_NAME_RESERVED_S1          0x50
1173 #define ACPI_RESOURCE_NAME_RESERVED_S2          0x58
1174 #define ACPI_RESOURCE_NAME_RESERVED_S3          0x60
1175 #define ACPI_RESOURCE_NAME_RESERVED_S4          0x68
1176 #define ACPI_RESOURCE_NAME_VENDOR_SMALL         0x70
1177 #define ACPI_RESOURCE_NAME_END_TAG              0x78

1179 /*
1180  * Large resource descriptor "names" as defined by the ACPI specification.
1181  * Note: includes the Large Descriptor bit in bit[7]
1182  */
1183 #define ACPI_RESOURCE_NAME_MEMORY24             0x81
1184 #define ACPI_RESOURCE_NAME_GENERIC_REGISTER     0x82
1185 #define ACPI_RESOURCE_NAME_RESERVED_L1          0x83
1186 #define ACPI_RESOURCE_NAME_VENDOR_LARGE         0x84
1187 #define ACPI_RESOURCE_NAME_MEMORY32             0x85
1188 #define ACPI_RESOURCE_NAME_FIXED_MEMORY32       0x86
1189 #define ACPI_RESOURCE_NAME_ADDRESS32            0x87
1190 #define ACPI_RESOURCE_NAME_ADDRESS16            0x88
1191 #define ACPI_RESOURCE_NAME_EXTENDED_IRQ         0x89
1192 #define ACPI_RESOURCE_NAME_ADDRESS64            0x8A
1193 #define ACPI_RESOURCE_NAME_EXTENDED_ADDRESS64   0x8B
1194 #define ACPI_RESOURCE_NAME_GPIO                 0x8C
1195 #define ACPI_RESOURCE_NAME_SERIAL_BUS           0x8E
1196 #define ACPI_RESOURCE_NAME_LARGE_MAX            0x8E
1137 #define ACPI_RESOURCE_NAME_LARGE_MAX            0x8B
```

```
/*****************************************************************************
 *
 * Miscellaneous
 *
 ****************************************************************************/

#define ACPI_ASCII_ZERO                 0x30


/*****************************************************************************
 *
 * Disassembler
 *
 ****************************************************************************/

typedef struct acpi_external_list
{
    char                        *Path;
    char                        *InternalPath;
    struct acpi_external_list   *Next;
    UINT32                      Value;
    UINT16                      Length;
    UINT16                      Flags;
    UINT8                       Type;
    UINT8                       Flags;

} ACPI_EXTERNAL_LIST;

/* Values for Flags field above */

#define ACPI_EXT_RESOLVED_REFERENCE         0x01    /* Object was resolved durin
#define ACPI_EXT_ORIGIN_FROM_FILE           0x02    /* External came from a file
#define ACPI_EXT_INTERNAL_PATH_ALLOCATED    0x04    /* Deallocate internal path
#define ACPI_EXT_EXTERNAL_EMITTED           0x08    /* External() statement has
#define ACPI_IPATH_ALLOCATED    0x01


typedef struct acpi_external_file
{
    char                        *Path;
    struct acpi_external_file   *Next;

} ACPI_EXTERNAL_FILE;


/*****************************************************************************
 *
 * Debugger
 *
 ****************************************************************************/

typedef struct acpi_db_method_info
{
    ACPI_HANDLE                 Method;
    ACPI_HANDLE                 MainThreadGate;
    ACPI_HANDLE                 ThreadCompleteGate;
    ACPI_HANDLE                 InfoGate;
    ACPI_THREAD_ID              *Threads;
    UINT32                      NumThreads;
    UINT32                      NumCreated;
    UINT32                      NumCompleted;

    char                        *Name;
    UINT32                      Flags;
    UINT32                      NumLoops;
    char                        Pathname[ACPI_DB_LINE_BUFFER_SIZE];
```



```
    char                        Pathname[128];
    char                        **Args;
    ACPI_OBJECT_TYPE            *Types;

    /*
     * Arguments to be passed to method for the command
     * Threads -
     *   the Number of threads, ID of current thread and
     *   Index of current thread inside all them created.
     */
    char                        InitArgs;
    ACPI_OBJECT_TYPE            ArgTypes[4];
    char                        *Arguments[4];
    char                        NumThreadsStr[11];
    char                        IdOfThreadStr[11];
    char                        IndexOfThreadStr[11];

} ACPI_DB_METHOD_INFO;
______unchanged_portion_omitted_


#define ACPI_DB_DISABLE_OUTPUT          0x00
#define ACPI_DB_REDIRECTABLE_OUTPUT     0x01
#define ACPI_DB_CONSOLE_OUTPUT          0x02
#define ACPI_DB_DUPLICATE_OUTPUT        0x03


/*****************************************************************************
 *
 * Debug
 *
 ****************************************************************************/

/* Entry for a memory allocation (debug only) */

#define ACPI_MEM_MALLOC                 0
#define ACPI_MEM_CALLOC                 1
#define ACPI_MAX_MODULE_NAME            16

#define ACPI_COMMON_DEBUG_MEM_HEADER \
    struct acpi_debug_mem_block     *Previous; \
    struct acpi_debug_mem_block     *Next; \
    UINT32                          Size; \
    UINT32                          Component; \
    UINT32                          Line; \
    char                            Module[ACPI_MAX_MODULE_NAME]; \
    UINT8                           AllocType;

typedef struct acpi_debug_mem_header
{
    ACPI_COMMON_DEBUG_MEM_HEADER

} ACPI_DEBUG_MEM_HEADER;
______unchanged_portion_omitted_


#define ACPI_MEM_LIST_GLOBAL            0
#define ACPI_MEM_LIST_NSNODE            1
#define ACPI_MEM_LIST_MAX               1
#define ACPI_NUM_MEM_LISTS              2


/*****************************************************************************
 *
 * Info/help support
 *
```

```
1340  *****************************************************************************/

1342 typedef struct ah_predefined_name
1343 {
1344     char            *Name;
1345     char            *Description;
1346 #ifndef ACPI_ASL_COMPILER
1347     char            *Action;
1348 #endif

1350 } AH_PREDEFINED_NAME;

1352 #endif /* __ACLOCAL_H__ */
```

```
**********************************************************
   19650 Thu Dec 26 13:49:50 2013
new/usr/src/common/acpica/include/acmacros.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acmacros.h - C macros for the entire subsystem.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACMACROS_H__
#define __ACMACROS_H__


/*
 * Extract data using a pointer. Any more than a byte and we
 * get into potential aligment issues -- see the STORE macros below.
 * Use with care.
 */
#define ACPI_CAST8(ptr)                 ACPI_CAST_PTR (UINT8, (ptr))
#define ACPI_CAST16(ptr)                ACPI_CAST_PTR (UINT16, (ptr))
#define ACPI_CAST32(ptr)                ACPI_CAST_PTR (UINT32, (ptr))
#define ACPI_CAST64(ptr)                ACPI_CAST_PTR (UINT64, (ptr))
#define ACPI_GET8(ptr)                  (*ACPI_CAST8 (ptr))
#define ACPI_GET16(ptr)                 (*ACPI_CAST16 (ptr))
#define ACPI_GET32(ptr)                 (*ACPI_CAST32 (ptr))
```



```
#define ACPI_GET64(ptr)                 (*ACPI_CAST64 (ptr))
#define ACPI_SET8(ptr, val)             (*ACPI_CAST8 (ptr) = (UINT8) (val))
#define ACPI_SET16(ptr, val)            (*ACPI_CAST16 (ptr) = (UINT16) (val))
#define ACPI_SET32(ptr, val)            (*ACPI_CAST32 (ptr) = (UINT32) (val))
#define ACPI_SET64(ptr, val)            (*ACPI_CAST64 (ptr) = (UINT64) (val))

/*
 * printf() format helpers
 */

/* Split 64-bit integer into two 32-bit values. Use with %8.8X%8.8X */

#define ACPI_FORMAT_UINT64(i)           ACPI_HIDWORD(i), ACPI_LODWORD(i)

#if ACPI_MACHINE_WIDTH == 64
#define ACPI_FORMAT_NATIVE_UINT(i)      ACPI_FORMAT_UINT64(i)
#else
#define ACPI_FORMAT_NATIVE_UINT(i)      0, (i)
#endif


/*
 * Macros for moving data around to/from buffers that are possibly unaligned.
 * If the hardware supports the transfer of unaligned data, just do the store.
 * Otherwise, we have to move one byte at a time.
 */
#ifdef ACPI_BIG_ENDIAN
/*
 * Macros for big-endian machines
 */

/* These macros reverse the bytes during the move, converting little-endian to b

                                                    /* Big Endian      <==
                                                    /*  Hi...Lo
/* 16-bit source, 16/32/64 destination */

#define ACPI_MOVE_16_TO_16(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *

#define ACPI_MOVE_16_TO_32(d, s)        {(*(UINT32 *)(void *)(d))=0;\
                                            ((UINT8 *)(void *)(d))[2] = ((UINT8 *
                                            ((UINT8 *)(void *)(d))[3] = ((UINT8 *

#define ACPI_MOVE_16_TO_64(d, s)        {(*(UINT64 *)(void *)(d))=0;\
                                            ((UINT8 *)(void *)(d))[6] = ((UINT8 *
                                            ((UINT8 *)(void *)(d))[7] = ((UINT8 *

/* 32-bit source, 16/32/64 destination */

#define ACPI_MOVE_32_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate

#define ACPI_MOVE_32_TO_32(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[2] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[3] = ((UINT8 *

#define ACPI_MOVE_32_TO_64(d, s)        {(*(UINT64 *)(void *)(d))=0;\
                                            ((UINT8 *)(void *)(d))[4] = ((UINT8 *
                                            ((UINT8 *)(void *)(d))[5] = ((UINT8 *
                                            ((UINT8 *)(void *)(d))[6] = ((UINT8 *
                                            ((UINT8 *)(void *)(d))[7] = ((UINT8 *

/* 64-bit source, 16/32/64 destination */

#define ACPI_MOVE_64_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate
```

```
#define ACPI_MOVE_64_TO_32(d, s)        ACPI_MOVE_32_TO_32(d, s)    /* Truncate

#define ACPI_MOVE_64_TO_64(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[2] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[3] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[4] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[5] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[6] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[7] = ((UINT8 *
#else
/*
 * Macros for little-endian machines
 */

#ifndef ACPI_MISALIGNMENT_NOT_SUPPORTED

/* The hardware supports unaligned transfers, just do the little-endian move */

/* 16-bit source, 16/32/64 destination */

#define ACPI_MOVE_16_TO_16(d, s)        *(UINT16 *)(void *)(d) = *(UINT16 *)(voi
#define ACPI_MOVE_16_TO_32(d, s)        *(UINT32 *)(void *)(d) = *(UINT16 *)(voi
#define ACPI_MOVE_16_TO_64(d, s)        *(UINT64 *)(void *)(d) = *(UINT16 *)(voi

/* 32-bit source, 16/32/64 destination */

#define ACPI_MOVE_32_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate
#define ACPI_MOVE_32_TO_32(d, s)        *(UINT32 *)(void *)(d) = *(UINT32 *)(voi
#define ACPI_MOVE_32_TO_64(d, s)        *(UINT64 *)(void *)(d) = *(UINT32 *)(voi

/* 64-bit source, 16/32/64 destination */

#define ACPI_MOVE_64_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate
#define ACPI_MOVE_64_TO_32(d, s)        ACPI_MOVE_32_TO_32(d, s)    /* Truncate
#define ACPI_MOVE_64_TO_64(d, s)        *(UINT64 *)(void *)(d) = *(UINT64 *)(voi

#else
/*
 * The hardware does not support unaligned transfers. We must move the
 * data one byte at a time. These macros work whether the source or
 * the destination (or both) is/are unaligned. (Little-endian move)
 */

/* 16-bit source, 16/32/64 destination */

#define ACPI_MOVE_16_TO_16(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *

#define ACPI_MOVE_16_TO_32(d, s)        {(*(UINT32 *)(void *)(d)) = 0; ACPI_MOVE
#define ACPI_MOVE_16_TO_64(d, s)        {(*(UINT64 *)(void *)(d)) = 0; ACPI_MOVE

/* 32-bit source, 16/32/64 destination */

#define ACPI_MOVE_32_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate

#define ACPI_MOVE_32_TO_32(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[2] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[3] = ((UINT8 *

#define ACPI_MOVE_32_TO_64(d, s)        {(*(UINT64 *)(void *)(d)) = 0; ACPI_MOVE

/* 64-bit source, 16/32/64 destination */
```



```
#define ACPI_MOVE_64_TO_16(d, s)        ACPI_MOVE_16_TO_16(d, s)    /* Truncate
#define ACPI_MOVE_64_TO_32(d, s)        ACPI_MOVE_32_TO_32(d, s)    /* Truncate
#define ACPI_MOVE_64_TO_64(d, s)        {((  UINT8 *)(void *)(d))[0] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[1] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[2] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[3] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[4] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[5] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[6] = ((UINT8 *
                                         ((  UINT8 *)(void *)(d))[7] = ((UINT8 *
#endif
#endif


/*
 * Fast power-of-two math macros for non-optimized compilers
 */
#define _ACPI_DIV(value, PowerOf2)      ((UINT32) ((value) >> (PowerOf2)))
#define _ACPI_MUL(value, PowerOf2)      ((UINT32) ((value) << (PowerOf2)))
#define _ACPI_MOD(value, Divisor)       ((UINT32) ((value) & ((Divisor) -1)))

#define ACPI_DIV_2(a)                   _ACPI_DIV(a, 1)
#define ACPI_MUL_2(a)                   _ACPI_MUL(a, 1)
#define ACPI_MOD_2(a)                   _ACPI_MOD(a, 2)

#define ACPI_DIV_4(a)                   _ACPI_DIV(a, 2)
#define ACPI_MUL_4(a)                   _ACPI_MUL(a, 2)
#define ACPI_MOD_4(a)                   _ACPI_MOD(a, 4)

#define ACPI_DIV_8(a)                   _ACPI_DIV(a, 3)
#define ACPI_MUL_8(a)                   _ACPI_MUL(a, 3)
#define ACPI_MOD_8(a)                   _ACPI_MOD(a, 8)

#define ACPI_DIV_16(a)                  _ACPI_DIV(a, 4)
#define ACPI_MUL_16(a)                  _ACPI_MUL(a, 4)
#define ACPI_MOD_16(a)                  _ACPI_MOD(a, 16)

#define ACPI_DIV_32(a)                  _ACPI_DIV(a, 5)
#define ACPI_MUL_32(a)                  _ACPI_MUL(a, 5)
#define ACPI_MOD_32(a)                  _ACPI_MOD(a, 32)

/*
 * Rounding macros (Power of two boundaries only)
 */
#define ACPI_ROUND_DOWN(value, boundary)    (((ACPI_SIZE)(value)) & \
                                            (~(((ACPI_SIZE) boundary)-1)))

#define ACPI_ROUND_UP(value, boundary)      ((((ACPI_SIZE)(value)) + \
                                            (((ACPI_SIZE) boundary)-1)) & \
                                            (~(((ACPI_SIZE) boundary)-1)))

/* Note: sizeof(ACPI_SIZE) evaluates to either 4 or 8 (32- vs 64-bit mode) */

#define ACPI_ROUND_DOWN_TO_32BIT(a)         ACPI_ROUND_DOWN(a, 4)
#define ACPI_ROUND_DOWN_TO_64BIT(a)         ACPI_ROUND_DOWN(a, 8)
#define ACPI_ROUND_DOWN_TO_NATIVE_WORD(a)   ACPI_ROUND_DOWN(a, sizeof(ACPI_SIZE)

#define ACPI_ROUND_UP_TO_32BIT(a)           ACPI_ROUND_UP(a, 4)
#define ACPI_ROUND_UP_TO_64BIT(a)           ACPI_ROUND_UP(a, 8)
#define ACPI_ROUND_UP_TO_NATIVE_WORD(a)     ACPI_ROUND_UP(a, sizeof(ACPI_SIZE))

#define ACPI_ROUND_BITS_UP_TO_BYTES(a)      ACPI_DIV_8((a) + 7)
#define ACPI_ROUND_BITS_DOWN_TO_BYTES(a)    ACPI_DIV_8((a))

#define ACPI_ROUND_UP_TO_1K(a)              (((a) + 1023) >> 10)
```

```
258 /* Generic (non-power-of-two) rounding */

260 #define ACPI_ROUND_UP_TO(value, boundary)   (((value) + ((boundary)-1)) / (bound

262 #define ACPI_IS_MISALIGNED(value)           (((ACPI_SIZE) value) & (sizeof(ACPI_

264 /*
265  * Bitmask creation
266  * Bit positions start at zero.
267  * MASK_BITS_ABOVE creates a mask starting AT the position and above
268  * MASK_BITS_BELOW creates a mask starting one bit BELOW the position
269  */
270 #define ACPI_MASK_BITS_ABOVE(position)      (~((ACPI_UINT64_MAX) << ((UINT32) (p
271 #define ACPI_MASK_BITS_BELOW(position)      ((ACPI_UINT64_MAX) << ((UINT32) (pos

273 /* Bitfields within ACPI registers */

275 #define ACPI_REGISTER_PREPARE_BITS(Val, Pos, Mask) \
276     ((Val << Pos) & Mask)

278 #define ACPI_REGISTER_INSERT_VALUE(Reg, Pos, Mask, Val) \
279     Reg = (Reg & (~(Mask))) | ACPI_REGISTER_PREPARE_BITS(Val, Pos, Mask)

281 #define ACPI_INSERT_BITS(Target, Mask, Source) \
282     Target = ((Target & (~(Mask))) | (Source & Mask))

284 /* Generic bitfield macros and masks */

286 #define ACPI_GET_BITS(SourcePtr, Position, Mask) \
287     ((*SourcePtr >> Position) & Mask)

289 #define ACPI_SET_BITS(TargetPtr, Position, Mask, Value) \
290     (*TargetPtr |= ((Value & Mask) << Position))

292 #define ACPI_1BIT_MASK      0x00000001
293 #define ACPI_2BIT_MASK      0x00000003
294 #define ACPI_3BIT_MASK      0x00000007
295 #define ACPI_4BIT_MASK      0x0000000F
296 #define ACPI_5BIT_MASK      0x0000001F
297 #define ACPI_6BIT_MASK      0x0000003F
298 #define ACPI_7BIT_MASK      0x0000007F
299 #define ACPI_8BIT_MASK      0x000000FF
300 #define ACPI_16BIT_MASK     0x0000FFFF
301 #define ACPI_24BIT_MASK     0x00FFFFFF

303 /* Macros to extract flag bits from position zero */

305 #define ACPI_GET_1BIT_FLAG(Value)                   ((Value) & ACPI_1BIT_MASK)
306 #define ACPI_GET_2BIT_FLAG(Value)                   ((Value) & ACPI_2BIT_MASK)
307 #define ACPI_GET_3BIT_FLAG(Value)                   ((Value) & ACPI_3BIT_MASK)
308 #define ACPI_GET_4BIT_FLAG(Value)                   ((Value) & ACPI_4BIT_MASK)

310 /* Macros to extract flag bits from position one and above */

312 #define ACPI_EXTRACT_1BIT_FLAG(Field, Position)     (ACPI_GET_1BIT_FLAG ((Field)
313 #define ACPI_EXTRACT_2BIT_FLAG(Field, Position)     (ACPI_GET_2BIT_FLAG ((Field)
314 #define ACPI_EXTRACT_3BIT_FLAG(Field, Position)     (ACPI_GET_3BIT_FLAG ((Field)
315 #define ACPI_EXTRACT_4BIT_FLAG(Field, Position)     (ACPI_GET_4BIT_FLAG ((Field)

317 /* ACPI Pathname helpers */

319 #define ACPI_IS_ROOT_PREFIX(c)      ((c) == (UINT8) 0x5C) /* Backslash */
320 #define ACPI_IS_PARENT_PREFIX(c)    ((c) == (UINT8) 0x5E) /* Carat */
321 #define ACPI_IS_PATH_SEPARATOR(c)   ((c) == (UINT8) 0x2E) /* Period (dot) */

323 /*
```



```
324  * An object of type ACPI_NAMESPACE_NODE can appear in some contexts
325  * where a pointer to an object of type ACPI_OPERAND_OBJECT can also
326  * appear. This macro is used to distinguish them.
327  *
328  * The "DescriptorType" field is the second field in both structures.
329  */
330 #define ACPI_GET_DESCRIPTOR_PTR(d)      (((ACPI_DESCRIPTOR *)(void *)(d))->Commo
331 #define ACPI_SET_DESCRIPTOR_PTR(d, p)   (((ACPI_DESCRIPTOR *)(void *)(d))->Commo
332 #define ACPI_GET_DESCRIPTOR_TYPE(d)     (((ACPI_DESCRIPTOR *)(void *)(d))->Commo
333 #define ACPI_SET_DESCRIPTOR_TYPE(d, t)  (((ACPI_DESCRIPTOR *)(void *)(d))->Commo

335 /*
336  * Macros for the master AML opcode table
337  */
338 #if defined (ACPI_DISASSEMBLER) || defined (ACPI_DEBUG_OUTPUT)
339 #define ACPI_OP(Name, PArgs, IArgs, ObjType, Class, Type, Flags) \
340     {Name, (UINT32)(PArgs), (UINT32)(IArgs), (UINT32)(Flags), ObjType, Class, Ty
341 #else
342 #define ACPI_OP(Name, PArgs, IArgs, ObjType, Class, Type, Flags) \
343     {(UINT32)(PArgs), (UINT32)(IArgs), (UINT32)(Flags), ObjType, Class, Type}
344 #endif

346 #define ARG_TYPE_WIDTH                  5
347 #define ARG_1(x)                        ((UINT32)(x))
348 #define ARG_2(x)                        ((UINT32)(x) << (1 * ARG_TYPE_WIDTH))
349 #define ARG_3(x)                        ((UINT32)(x) << (2 * ARG_TYPE_WIDTH))
350 #define ARG_4(x)                        ((UINT32)(x) << (3 * ARG_TYPE_WIDTH))
351 #define ARG_5(x)                        ((UINT32)(x) << (4 * ARG_TYPE_WIDTH))
352 #define ARG_6(x)                        ((UINT32)(x) << (5 * ARG_TYPE_WIDTH))

354 #define ARGI_LIST1(a)                   (ARG_1(a))
355 #define ARGI_LIST2(a, b)                (ARG_1(b)|ARG_2(a))
356 #define ARGI_LIST3(a, b, c)             (ARG_1(c)|ARG_2(b)|ARG_3(a))
357 #define ARGI_LIST4(a, b, c, d)          (ARG_1(d)|ARG_2(c)|ARG_3(b)|ARG_4(a))
358 #define ARGI_LIST5(a, b, c, d, e)       (ARG_1(e)|ARG_2(d)|ARG_3(c)|ARG_4(b)|ARG
359 #define ARGI_LIST6(a, b, c, d, e, f)    (ARG_1(f)|ARG_2(e)|ARG_3(d)|ARG_4(c)|ARG

361 #define ARGP_LIST1(a)                   (ARG_1(a))
362 #define ARGP_LIST2(a, b)                (ARG_1(a)|ARG_2(b))
363 #define ARGP_LIST3(a, b, c)             (ARG_1(a)|ARG_2(b)|ARG_3(c))
364 #define ARGP_LIST4(a, b, c, d)          (ARG_1(a)|ARG_2(b)|ARG_3(c)|ARG_4(d))
365 #define ARGP_LIST5(a, b, c, d, e)       (ARG_1(a)|ARG_2(b)|ARG_3(c)|ARG_4(d)|ARG
366 #define ARGP_LIST6(a, b, c, d, e, f)    (ARG_1(a)|ARG_2(b)|ARG_3(c)|ARG_4(d)|ARG

368 #define GET_CURRENT_ARG_TYPE(List)      (List & ((UINT32) 0x1F))
369 #define INCREMENT_ARG_LIST(List)        (List >>= ((UINT32) ARG_TYPE_WIDTH))

371 /*
372  * Ascii error messages can be configured out
373  */
374 #ifndef ACPI_NO_ERROR_MESSAGES
375 /*
376  * Error reporting. Callers module and line number are inserted by AE_INFO,
377  * the plist contains a set of parens to allow variable-length lists.
378  * These macros are used for both the debug and non-debug versions of the code.
379  */
380 #define ACPI_ERROR_NAMESPACE(s, e)          AcpiUtNamespaceError (AE_INFO, s, e)
381 #define ACPI_ERROR_METHOD(s, n, p, e)       AcpiUtMethodError (AE_INFO, s, n, p,
382 #define ACPI_WARN_PREDEFINED(plist)         AcpiUtPredefinedWarning plist
383 #define ACPI_INFO_PREDEFINED(plist)         AcpiUtPredefinedInfo plist
384 #define ACPI_BIOS_ERROR_PREDEFINED(plist)   AcpiUtPredefinedBiosError plist

386 #else

388 /* No error messages */
```

```
390 #define ACPI_ERROR_NAMESPACE(s, e)
391 #define ACPI_ERROR_METHOD(s, n, p, e)
392 #define ACPI_WARN_PREDEFINED(plist)
393 #define ACPI_INFO_PREDEFINED(plist)
394 #define ACPI_BIOS_ERROR_PREDEFINED(plist)

396 #endif /* ACPI_NO_ERROR_MESSAGES */

398 #if (!ACPI_REDUCED_HARDWARE)
399 #define ACPI_HW_OPTIONAL_FUNCTION(addr)     addr
400 #else
401 #define ACPI_HW_OPTIONAL_FUNCTION(addr)     NULL
402 #endif


405 /*
406  * Some code only gets executed when the debugger is built in.
407  * Note that this is entirely independent of whether the
408  * DEBUG_PRINT stuff (set by ACPI_DEBUG_OUTPUT) is on, or not.
409  */
410 #ifdef ACPI_DEBUGGER
411 #define ACPI_DEBUGGER_EXEC(a)           a
412 #else
413 #define ACPI_DEBUGGER_EXEC(a)
414 #endif


417 /*
418  * Macros used for ACPICA utilities only
419  */

421 /* Generate a UUID */

423 #define ACPI_INIT_UUID(a, b, c, d0, d1, d2, d3, d4, d5, d6, d7) \
424     (a) & 0xFF, ((a) >> 8) & 0xFF, ((a) >> 16) & 0xFF, ((a) >> 24) & 0xFF, \
425     (b) & 0xFF, ((b) >> 8) & 0xFF, \
426     (c) & 0xFF, ((c) >> 8) & 0xFF, \
427     (d0), (d1), (d2), (d3), (d4), (d5), (d6), (d7)

429 #define ACPI_IS_OCTAL_DIGIT(d)              (((char)(d) >= '0') && ((char)(d) <=


432 #endif /* ACMACROS_H */
```

```
**********************************************************
    3582 Thu Dec 26 13:49:50 2013
new/usr/src/common/acpica/include/acnames.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acnames.h - Global names and strings
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACNAMES_H__
#define __ACNAMES_H__

/* Method names - these methods can appear anywhere in the namespace */

#define METHOD_NAME__HID        "_HID"
#define METHOD_NAME__CID        "_CID"
#define METHOD_NAME__UID        "_UID"
#define METHOD_NAME__ADR        "_ADR"
#define METHOD_NAME__AEI        "_AEI"
#define METHOD_NAME__INI        "_INI"
#define METHOD_NAME__STA        "_STA"
#define METHOD_NAME__REG        "_REG"
#define METHOD_NAME__SEG        "_SEG"
#define METHOD_NAME__BBN        "_BBN"
```



```
#define METHOD_NAME__CBA        "_CBA"
#define METHOD_NAME__CID        "_CID"
#define METHOD_NAME__PRT        "_PRT"
#define METHOD_NAME__CRS        "_CRS"
#define METHOD_NAME__HID        "_HID"
#define METHOD_NAME__INI        "_INI"
#define METHOD_NAME__PLD        "_PLD"
#define METHOD_NAME__PRS        "_PRS"
#define METHOD_NAME__PRT        "_PRT"
#define METHOD_NAME__PRW        "_PRW"
#define METHOD_NAME__REG        "_REG"
#define METHOD_NAME__SB_        "_SB_"
#define METHOD_NAME__SEG        "_SEG"
#define METHOD_NAME__SRS        "_SRS"
#define METHOD_NAME__STA        "_STA"
#define METHOD_NAME__SUB        "_SUB"
#define METHOD_NAME__UID        "_UID"

/* Method names - these methods must appear at the namespace root */

#define METHOD_PATHNAME__PTS    "\\_PTS"
#define METHOD_PATHNAME__SST    "\\_SI._SST"
#define METHOD_PATHNAME__WAK    "\\_WAK"
#define METHOD_NAME__BFS        "\\_BFS"
#define METHOD_NAME__GTS        "\\_GTS"
#define METHOD_NAME__PTS        "\\_PTS"
#define METHOD_NAME__SST        "\\_SI._SST"
#define METHOD_NAME__WAK        "\\_WAK"

/* Definitions of the predefined namespace names  */

#define ACPI_UNKNOWN_NAME       (UINT32) 0x3F3F3F3F     /* Unknown name is "????
#define ACPI_ROOT_NAME          (UINT32) 0x5F5F5F5C     /* Root name is    "\___

#define ACPI_PREFIX_MIXED       (UINT32) 0x69706341     /* "Acpi" */
#define ACPI_PREFIX_LOWER       (UINT32) 0x69706361     /* "acpi" */

#define ACPI_NS_ROOT_PATH       "\\"
#define ACPI_NS_SYSTEM_BUS      "_SB_"

#endif  /* __ACNAMES_H__  */
```

```
**********************************************************
   13368 Thu Dec 26 13:49:51 2013
new/usr/src/common/acpica/include/acnamesp.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acnamesp.h - Namespace subcomponent prototypes and defines
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACNAMESP_H__
  45 #define __ACNAMESP_H__


  48 /* To search the entire name space, pass this as SearchBase */

  50 #define ACPI_NS_ALL                 ((ACPI_HANDLE)0)

  52 /*
  53  * Elements of AcpiNsProperties are bit significant
  54  * and should be one-to-one with values of ACPI_OBJECT_TYPE
  55  */
  56 #define ACPI_NS_NORMAL              0
  57 #define ACPI_NS_NEWSCOPE            1   /* a definition of this type opens a nam
  58 #define ACPI_NS_LOCAL               2   /* suppress search of enclosing scopes *
```



```
  60 /* Flags for AcpiNsLookup, AcpiNsSearchAndEnter */

  62 #define ACPI_NS_NO_UPSEARCH         0
  63 #define ACPI_NS_SEARCH_PARENT       0x01
  64 #define ACPI_NS_DONT_OPEN_SCOPE     0x02
  65 #define ACPI_NS_NO_PEER_SEARCH      0x04
  66 #define ACPI_NS_ERROR_IF_FOUND      0x08
  67 #define ACPI_NS_PREFIX_IS_SCOPE     0x10
  68 #define ACPI_NS_EXTERNAL            0x20
  69 #define ACPI_NS_TEMPORARY           0x40

  71 /* Flags for AcpiNsWalkNamespace */

  73 #define ACPI_NS_WALK_NO_UNLOCK      0
  74 #define ACPI_NS_WALK_UNLOCK         0x01
  75 #define ACPI_NS_WALK_TEMP_NODES     0x02

  77 /* Object is not a package element */

  79 #define ACPI_NOT_PACKAGE_ELEMENT    ACPI_UINT32_MAX

  81 /* Always emit warning message, not dependent on node flags */

  83 #define ACPI_WARN_ALWAYS            0


  86 /*
  87  * nsinit - Namespace initialization
  88  */
  89 ACPI_STATUS
  90 AcpiNsInitializeObjects (
  91     void);

  93 ACPI_STATUS
  94 AcpiNsInitializeDevices (
  95     void);


  98 /*
  99  * nsload -  Namespace loading
 100  */
 101 ACPI_STATUS
 102 AcpiNsLoadNamespace (
 103     void);

 105 ACPI_STATUS
 106 AcpiNsLoadTable (
 107     UINT32                  TableIndex,
 108     ACPI_NAMESPACE_NODE     *Node);


 111 /*
 112  * nswalk - walk the namespace
 113  */
 114 ACPI_STATUS
 115 AcpiNsWalkNamespace (
 116     ACPI_OBJECT_TYPE        Type,
 117     ACPI_HANDLE             StartObject,
 118     UINT32                  MaxDepth,
 119     UINT32                  Flags,
 120     ACPI_WALK_CALLBACK      DescendingCallback,
 121     ACPI_WALK_CALLBACK      AscendingCallback,
 120     ACPI_WALK_CALLBACK      PreOrderVisit,
 121     ACPI_WALK_CALLBACK      PostOrderVisit,
 122     void                    *Context,
 123     void                    **ReturnValue);
```

```
125 ACPI_NAMESPACE_NODE *
126 AcpiNsGetNextNode (
127     ACPI_NAMESPACE_NODE     *Parent,
128     ACPI_NAMESPACE_NODE     *Child);

130 ACPI_NAMESPACE_NODE *
131 AcpiNsGetNextNodeTyped (
132     ACPI_OBJECT_TYPE        Type,
133     ACPI_NAMESPACE_NODE     *Parent,
134     ACPI_NAMESPACE_NODE     *Child);

136 /*
137  * nsparse - table parsing
138  */
139 ACPI_STATUS
140 AcpiNsParseTable (
141     UINT32                  TableIndex,
142     ACPI_NAMESPACE_NODE     *StartNode);

144 ACPI_STATUS
145 AcpiNsOneCompleteParse (
146     UINT32                  PassNumber,
147     UINT32                  TableIndex,
148     ACPI_NAMESPACE_NODE     *StartNode);


151 /*
152  * nsaccess - Top-level namespace access
153  */
154 ACPI_STATUS
155 AcpiNsRootInitialize (
156     void);

158 ACPI_STATUS
159 AcpiNsLookup (
160     ACPI_GENERIC_STATE      *ScopeInfo,
161     char                    *Name,
162     ACPI_OBJECT_TYPE        Type,
163     ACPI_INTERPRETER_MODE   InterpreterMode,
164     UINT32                  Flags,
165     ACPI_WALK_STATE         *WalkState,
166     ACPI_NAMESPACE_NODE     **RetNode);


169 /*
170  * nsalloc - Named object allocation/deallocation
171  */
172 ACPI_NAMESPACE_NODE *
173 AcpiNsCreateNode (
174     UINT32                  Name);

176 void
177 AcpiNsDeleteNode (
178     ACPI_NAMESPACE_NODE     *Node);

180 void
181 AcpiNsRemoveNode (
182     ACPI_NAMESPACE_NODE     *Node);

184 void
185 AcpiNsDeleteNamespaceSubtree (
186     ACPI_NAMESPACE_NODE     *ParentHandle);

188 void
189 AcpiNsDeleteNamespaceByOwner (
```



```
190     ACPI_OWNER_ID           OwnerId);

192 void
193 AcpiNsDetachObject (
194     ACPI_NAMESPACE_NODE     *Node);

196 void
197 AcpiNsDeleteChildren (
198     ACPI_NAMESPACE_NODE     *Parent);

200 int
201 AcpiNsCompareNames (
202     char                    *Name1,
203     char                    *Name2);


206 /*
207  * nsconvert - Dynamic object conversion routines
208  */
209 ACPI_STATUS
210 AcpiNsConvertToInteger (
211     ACPI_OPERAND_OBJECT     *OriginalObject,
212     ACPI_OPERAND_OBJECT     **ReturnObject);

214 ACPI_STATUS
215 AcpiNsConvertToString (
216     ACPI_OPERAND_OBJECT     *OriginalObject,
217     ACPI_OPERAND_OBJECT     **ReturnObject);

219 ACPI_STATUS
220 AcpiNsConvertToBuffer (
221     ACPI_OPERAND_OBJECT     *OriginalObject,
222     ACPI_OPERAND_OBJECT     **ReturnObject);

224 ACPI_STATUS
225 AcpiNsConvertToUnicode (
226     ACPI_OPERAND_OBJECT     *OriginalObject,
227     ACPI_OPERAND_OBJECT     **ReturnObject);

229 ACPI_STATUS
230 AcpiNsConvertToResource (
231     ACPI_OPERAND_OBJECT     *OriginalObject,
232     ACPI_OPERAND_OBJECT     **ReturnObject);


235 /*
236  * nsdump - Namespace dump/print utilities
237  */
238 void
239 AcpiNsDumpTables (
240     ACPI_HANDLE             SearchBase,
241     UINT32                  MaxDepth);

243 void
244 AcpiNsDumpEntry (
245     ACPI_HANDLE             Handle,
246     UINT32                  DebugLevel);

248 void
249 AcpiNsDumpPathname (
250     ACPI_HANDLE             Handle,
251     char                    *Msg,
252     UINT32                  Level,
253     UINT32                  Component);

255 void
```

```
256 AcpiNsPrintPathname (
257     UINT32                  NumSegments,
258     char                    *Pathname);

260 ACPI_STATUS
261 AcpiNsDumpOneObject (
262     ACPI_HANDLE             ObjHandle,
263     UINT32                  Level,
264     void                    *Context,
265     void                    **ReturnValue);

267 void
268 AcpiNsDumpObjects (
269     ACPI_OBJECT_TYPE        Type,
270     UINT8                   DisplayType,
271     UINT32                  MaxDepth,
272     ACPI_OWNER_ID           OwnerId,
273     ACPI_HANDLE             StartHandle);

275 void
276 AcpiNsDumpObjectPaths (
277     ACPI_OBJECT_TYPE        Type,
278     UINT8                   DisplayType,
279     UINT32                  MaxDepth,
280     ACPI_OWNER_ID           OwnerId,
281     ACPI_HANDLE             StartHandle);


284 /*
285  * nseval - Namespace evaluation functions
286  */
287 ACPI_STATUS
288 AcpiNsEvaluate (
289     ACPI_EVALUATE_INFO      *Info);

291 void
292 AcpiNsExecModuleCodeList (
293     void);


296 /*
297  * nsarguments - Argument count/type checking for predefined/reserved names
260  * nspredef - Support for predefined/reserved names
298  */
299 void
300 AcpiNsCheckArgumentCount (
301     char                    *Pathname,
262 ACPI_STATUS
263 AcpiNsCheckPredefinedNames (
302     ACPI_NAMESPACE_NODE     *Node,
303     UINT32                  UserParamCount,
304     const ACPI_PREDEFINED_INFO  *Info);
266     ACPI_STATUS             ReturnStatus,
267     ACPI_OPERAND_OBJECT     **ReturnObject);

269 const ACPI_PREDEFINED_INFO *
270 AcpiNsCheckForPredefinedName (
271     ACPI_NAMESPACE_NODE     *Node);

306 void
307 AcpiNsCheckAcpiCompliance (
274 AcpiNsCheckParameterCount (
308     char                    *Pathname,
309     ACPI_NAMESPACE_NODE     *Node,
310     const ACPI_PREDEFINED_INFO  *Predefined);
```



```
312 void
313 AcpiNsCheckArgumentTypes (
314     ACPI_EVALUATE_INFO      *Info);


317 /*
318  * nspredef - Return value checking for predefined/reserved names
319  */
320 ACPI_STATUS
321 AcpiNsCheckReturnValue (
322     ACPI_NAMESPACE_NODE     *Node,
323     ACPI_EVALUATE_INFO      *Info,
324     UINT32                  UserParamCount,
325     ACPI_STATUS             ReturnStatus,
326     ACPI_OPERAND_OBJECT     **ReturnObject);
278     const ACPI_PREDEFINED_INFO  *Info);

328 ACPI_STATUS
329 AcpiNsCheckObjectType (
330     ACPI_EVALUATE_INFO      *Info,
331     ACPI_OPERAND_OBJECT     **ReturnObjectPtr,
332     UINT32                  ExpectedBtypes,
333     UINT32                  PackageIndex);


336 /*
337  * nsprepkg - Validation of predefined name packages
338  */
339 ACPI_STATUS
340 AcpiNsCheckPackage (
341     ACPI_EVALUATE_INFO      *Info,
342     ACPI_OPERAND_OBJECT     **ReturnObjectPtr);


345 /*
346  * nsnames - Name and Scope manipulation
347  */
348 UINT32
349 AcpiNsOpensScope (
350     ACPI_OBJECT_TYPE        Type);

352 ACPI_STATUS
353 AcpiNsBuildExternalPath (
354     ACPI_NAMESPACE_NODE     *Node,
355     ACPI_SIZE               Size,
356     char                    *NameBuffer);

358 char *
359 AcpiNsGetExternalPathname (
360     ACPI_NAMESPACE_NODE     *Node);

362 char *
363 AcpiNsNameOfCurrentScope (
364     ACPI_WALK_STATE         *WalkState);

366 ACPI_STATUS
367 AcpiNsHandleToPathname (
368     ACPI_HANDLE             TargetHandle,
369     ACPI_BUFFER             *Buffer);

371 BOOLEAN
372 AcpiNsPatternMatch (
373     ACPI_NAMESPACE_NODE     *ObjNode,
374     char                    *SearchFor);

376 ACPI_STATUS
```

```
377 AcpiNsGetNode (
378     ACPI_NAMESPACE_NODE     *PrefixNode,
379     const char              *ExternalPathname,
380     UINT32                  Flags,
381     ACPI_NAMESPACE_NODE     **OutNode);

383 ACPI_SIZE
384 AcpiNsGetPathnameLength (
385     ACPI_NAMESPACE_NODE     *Node);


388 /*
389  * nsobject - Object management for namespace nodes
390  */
391 ACPI_STATUS
392 AcpiNsAttachObject (
393     ACPI_NAMESPACE_NODE     *Node,
394     ACPI_OPERAND_OBJECT     *Object,
395     ACPI_OBJECT_TYPE        Type);

397 ACPI_OPERAND_OBJECT *
398 AcpiNsGetAttachedObject (
399     ACPI_NAMESPACE_NODE     *Node);

401 ACPI_OPERAND_OBJECT *
402 AcpiNsGetSecondaryObject (
403     ACPI_OPERAND_OBJECT     *ObjDesc);

405 ACPI_STATUS
406 AcpiNsAttachData (
407     ACPI_NAMESPACE_NODE     *Node,
408     ACPI_OBJECT_HANDLER     Handler,
409     void                    *Data);

411 ACPI_STATUS
412 AcpiNsDetachData (
413     ACPI_NAMESPACE_NODE     *Node,
414     ACPI_OBJECT_HANDLER     Handler);

416 ACPI_STATUS
417 AcpiNsGetAttachedData (
418     ACPI_NAMESPACE_NODE     *Node,
419     ACPI_OBJECT_HANDLER     Handler,
420     void                    **Data);


423 /*
424  * nsrepair - General return object repair for all
425  * predefined methods/objects
426  */
427 ACPI_STATUS
428 AcpiNsSimpleRepair (
429     ACPI_EVALUATE_INFO      *Info,
364 AcpiNsRepairObject (
365     ACPI_PREDEFINED_DATA    *Data,
430     UINT32                  ExpectedBtypes,
431     UINT32                  PackageIndex,
432     ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

434 ACPI_STATUS
435 AcpiNsWrapWithPackage (
436     ACPI_EVALUATE_INFO      *Info,
437     ACPI_OPERAND_OBJECT     *OriginalObject,
371 AcpiNsRepairPackageList (
372     ACPI_PREDEFINED_DATA    *Data,
438     ACPI_OPERAND_OBJECT     **ObjDescPtr);
```



```
440 ACPI_STATUS
441 AcpiNsRepairNullElement (
442     ACPI_EVALUATE_INFO      *Info,
377     ACPI_PREDEFINED_DATA    *Data,
443     UINT32                  ExpectedBtypes,
444     UINT32                  PackageIndex,
445     ACPI_OPERAND_OBJECT     **ReturnObjectPtr);

447 void
448 AcpiNsRemoveNullElements (
449     ACPI_EVALUATE_INFO      *Info,
384     ACPI_PREDEFINED_DATA    *Data,
450     UINT8                   PackageType,
451     ACPI_OPERAND_OBJECT     *ObjDesc);


454 /*
455  * nsrepair2 - Return object repair for specific
456  * predefined methods/objects
457  */
458 ACPI_STATUS
459 AcpiNsComplexRepairs (
460     ACPI_EVALUATE_INFO      *Info,
395     ACPI_PREDEFINED_DATA    *Data,
461     ACPI_NAMESPACE_NODE     *Node,
462     ACPI_STATUS             ValidateStatus,
463     ACPI_OPERAND_OBJECT     **ReturnObjectPtr);


466 /*
467  * nssearch - Namespace searching and entry
468  */
469 ACPI_STATUS
470 AcpiNsSearchAndEnter (
471     UINT32                  EntryName,
472     ACPI_WALK_STATE         *WalkState,
473     ACPI_NAMESPACE_NODE     *Node,
474     ACPI_INTERPRETER_MODE   InterpreterMode,
475     ACPI_OBJECT_TYPE        Type,
476     UINT32                  Flags,
477     ACPI_NAMESPACE_NODE     **RetNode);

479 ACPI_STATUS
480 AcpiNsSearchOneScope (
481     UINT32                  EntryName,
482     ACPI_NAMESPACE_NODE     *Node,
483     ACPI_OBJECT_TYPE        Type,
484     ACPI_NAMESPACE_NODE     **RetNode);

486 void
487 AcpiNsInstallNode (
488     ACPI_WALK_STATE         *WalkState,
489     ACPI_NAMESPACE_NODE     *ParentNode,
490     ACPI_NAMESPACE_NODE     *Node,
491     ACPI_OBJECT_TYPE        Type);


494 /*
495  * nsutils - Utility functions
496  */
432 BOOLEAN
433 AcpiNsValidRootPrefix (
434     char                    Prefix);

497 ACPI_OBJECT_TYPE
```

```
498 AcpiNsGetType (
499     ACPI_NAMESPACE_NODE     *Node);

501 UINT32
502 AcpiNsLocal (
503     ACPI_OBJECT_TYPE        Type);

505 void
506 AcpiNsPrintNodePathname (
507     ACPI_NAMESPACE_NODE     *Node,
508     const char              *Msg);

510 ACPI_STATUS
511 AcpiNsBuildInternalName (
512     ACPI_NAMESTRING_INFO    *Info);

514 void
515 AcpiNsGetInternalNameLength (
516     ACPI_NAMESTRING_INFO    *Info);

518 ACPI_STATUS
519 AcpiNsInternalizeName (
520     const char              *DottedName,
521     char                    **ConvertedName);

523 ACPI_STATUS
524 AcpiNsExternalizeName (
525     UINT32                  InternalNameLength,
526     const char              *InternalName,
527     UINT32                  *ConvertedNameLength,
528     char                    **ConvertedName);

530 ACPI_NAMESPACE_NODE *
531 AcpiNsValidateHandle (
532     ACPI_HANDLE             Handle);

534 void
535 AcpiNsTerminate (
536     void);

538 #endif /* __ACNAMESP_H__ */
```

****
21194 Thu Dec 26 13:49:51 2013
new/usr/src/common/acpica/include/acobject.h
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acobject.h - Definition of ACPI_OPERAND_OBJECT  (Internal object only)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _ACOBJECT_H
#define _ACOBJECT_H

/* acpisrc:StructDefs -- for acpisrc conversion */


/*
 * The ACPI_OPERAND_OBJECT is used to pass AML operands from the dispatcher
 * to the interpreter, and to keep track of the various handlers such as
 * address space handlers and notify handlers. The object is a constant
 * size in order to allow it to be cached and reused.
 *
 * Note: The object is optimized to be aligned and will not work if it is
 * byte-packed.
 */
```



```
#if ACPI_MACHINE_WIDTH == 64
#pragma pack(8)
#else
#pragma pack(4)
#endif

/*****************************************************************************
 *
 * Common Descriptors
 *
 ****************************************************************************/

/*
 * Common area for all objects.
 *
 * DescriptorType is used to differentiate between internal descriptors, and
 * must be in the same place across all descriptors
 *
 * Note: The DescriptorType and Type fields must appear in the identical
 * position in both the ACPI_NAMESPACE_NODE and ACPI_OPERAND_OBJECT
 * structures.
 */
#define ACPI_OBJECT_COMMON_HEADER \
    union acpi_operand_object       *NextObject;        /* Objects linked to par
    UINT8                           DescriptorType;     /* To differentiate vari
    UINT8                           Type;               /* ACPI_OBJECT_TYPE */\
    UINT16                          ReferenceCount;     /* For object deletion m
    UINT8                           Flags;
    /*
     * Note: There are 3 bytes available here before the
     * next natural alignment boundary (for both 32/64 cases)
     */

/* Values for Flag byte above */

#define AOPOBJ_AML_CONSTANT         0x01    /* Integer is an AML constant */
#define AOPOBJ_STATIC_POINTER       0x02    /* Data is part of an ACPI table, do
#define AOPOBJ_DATA_VALID           0x04    /* Object is initialized and data is
#define AOPOBJ_DATA_VALID           0x04    /* Object is intialized and data is
#define AOPOBJ_OBJECT_INITIALIZED   0x08    /* Region is initialized, _REG was r
#define AOPOBJ_SETUP_COMPLETE       0x10    /* Region setup is complete */
#define AOPOBJ_INVALID              0x20    /* Host OS won't allow a Region addr


/*****************************************************************************
 *
 * Basic data types
 *
 ****************************************************************************/

typedef struct acpi_object_common
{
    ACPI_OBJECT_COMMON_HEADER

} ACPI_OBJECT_COMMON;
```

unchanged_portion_omitted_

```
/*
 * Note: The String and Buffer object must be identical through the
 * pointer and length elements. There is code that depends on this.
 * Note: The String and Buffer object must be identical through the Pointer
 * and length elements.  There is code that depends on this.
 *
 * Fields common to both Strings and Buffers
 */
```

```
130 #define ACPI_COMMON_BUFFER_INFO(_Type) \
131     _Type                       *Pointer; \
132     UINT32                      Length;


135 typedef struct acpi_object_string   /* Null terminated, ASCII characters only */
136 {
137     ACPI_OBJECT_COMMON_HEADER
138     ACPI_COMMON_BUFFER_INFO         (char)              /* String in AML stream

140 } ACPI_OBJECT_STRING;
```

*unchanged_portion_omitted_*

```
229 /* Flags for InfoFlags field above */

231 #define ACPI_METHOD_MODULE_LEVEL        0x01    /* Method is actually module-lev
232 #define ACPI_METHOD_INTERNAL_ONLY       0x02    /* Method is implemented interna
233 #define ACPI_METHOD_SERIALIZED          0x04    /* Method is serialized */
234 #define ACPI_METHOD_SERIALIZED_PENDING  0x08    /* Method is to be marked serial
235 #define ACPI_METHOD_MODIFIED_NAMESPACE  0x10    /* Method modified the namespace


238 /*****************************************************************************
239  *
240  * Objects that can be notified. All share a common NotifyInfo area.
241  *
242  ****************************************************************************/

244 /*
245  * Common fields for objects that support ASL notifications
246  */
247 #define ACPI_COMMON_NOTIFY_INFO \
248     union acpi_operand_object       *NotifyList[2];     /* Handlers for system/d
249     union acpi_operand_object       *SystemNotify;      /* Handler for system no
250     union acpi_operand_object       *DeviceNotify;      /* Handler for driver no
249     union acpi_operand_object       *Handler;           /* Handler for Address s


252 typedef struct acpi_object_notify_common    /* COMMON NOTIFY for POWER, PROCESSO
253 {
254     ACPI_OBJECT_COMMON_HEADER
255     ACPI_COMMON_NOTIFY_INFO

257 } ACPI_OBJECT_NOTIFY_COMMON;
```

*unchanged_portion_omitted_*

```
301 /*****************************************************************************
302  *
303  * Fields. All share a common header/info field.
304  *
305  ****************************************************************************/

307 /*
308  * Common bitfield for the field objects
309  * "Field Datum"  -- a datum from the actual field object
310  * "Buffer Datum" -- a datum from a user buffer, read from or to be written to t
311  */
312 #define ACPI_COMMON_FIELD_INFO \
313     UINT8                       FieldFlags;         /* Access, update, and l
314     UINT8                       Attribute;          /* From AccessAs keyword
315     UINT8                       AccessByteWidth;    /* Read/Write size in by
316     ACPI_NAMESPACE_NODE         *Node;              /* Link back to parent n
317     UINT32                      BitLength;          /* Length of field in bi
318     UINT32                      BaseByteOffset;     /* Byte offset within co
319     UINT32                      Value;              /* Value to store into t
```



```
320     UINT8                       StartFieldBitOffset;/* Bit offset within fir
321     UINT8                       AccessLength;       /* For serial regions/fi


324 typedef struct acpi_object_field_common              /* COMMON FIELD (for BUF
325 {
326     ACPI_OBJECT_COMMON_HEADER
327     ACPI_COMMON_FIELD_INFO
328     union acpi_operand_object   *RegionObj;         /* Parent Operation Regi

330 } ACPI_OBJECT_FIELD_COMMON;


333 typedef struct acpi_object_region_field
334 {
335     ACPI_OBJECT_COMMON_HEADER
336     ACPI_COMMON_FIELD_INFO
337     UINT16                      ResourceLength;
338     union acpi_operand_object   *RegionObj;         /* Containing OpRegion o
339     UINT8                       *ResourceBuffer;    /* ResourceTemplate for

341 } ACPI_OBJECT_REGION_FIELD;
```

*unchanged_portion_omitted_*

```
380 /*****************************************************************************
381  *
382  * Objects for handlers
383  *
384  ****************************************************************************/

386 typedef struct acpi_object_notify_handler
387 {
388     ACPI_OBJECT_COMMON_HEADER
389     ACPI_NAMESPACE_NODE         *Node;              /* Parent device */
390     UINT32                      HandlerType;        /* Type: Device/System/B
391     ACPI_NOTIFY_HANDLER         Handler;            /* Handler address */
389     ACPI_NOTIFY_HANDLER         Handler;
392     void                        *Context;
393     union acpi_operand_object   *Next[2];           /* Device and System han

395 } ACPI_OBJECT_NOTIFY_HANDLER;


398 typedef struct acpi_object_addr_handler
399 {
400     ACPI_OBJECT_COMMON_HEADER
401     UINT8                       SpaceId;
402     UINT8                       HandlerFlags;
403     ACPI_ADR_SPACE_HANDLER      Handler;
404     ACPI_NAMESPACE_NODE         *Node;              /* Parent device */
405     void                        *Context;
406     ACPI_ADR_SPACE_SETUP        Setup;
407     union acpi_operand_object   *RegionList;        /* Regions using this ha
404     union acpi_operand_object   *RegionList;        /* regions using this ha
408     union acpi_operand_object   *Next;

410 } ACPI_OBJECT_ADDR_HANDLER;
```

*unchanged_portion_omitted_*

```
458 /*
459  * Extra object is used as additional storage for types that
460  * have AML code in their declarations (TermArgs) that must be
461  * evaluated at run time.
462  *
```

```
 463  * Currently: Region and FieldUnit types
 464  */
 465 typedef struct acpi_object_extra
 466 {
 467     ACPI_OBJECT_COMMON_HEADER
 468     ACPI_NAMESPACE_NODE             *Method_REG;        /* _REG method for this
 469     ACPI_NAMESPACE_NODE             *ScopeNode;
 470     void                            *RegionContext;     /* Region-specific data
 471     UINT8                           *AmlStart;
 472     UINT32                          AmlLength;

 474 } ACPI_OBJECT_EXTRA;
```

*unchanged_portion_omitted_*

```
**********************************************************
   22618 Thu Dec 26 13:49:52 2013
new/usr/src/common/acpica/include/acopcode.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acopcode.h - AML opcode information for the AML parser and interpreter
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACOPCODE_H__
  45 #define __ACOPCODE_H__

  47 #define MAX_EXTENDED_OPCODE         0x88
  48 #define NUM_EXTENDED_OPCODE         (MAX_EXTENDED_OPCODE + 1)
  49 #define MAX_INTERNAL_OPCODE
  50 #define NUM_INTERNAL_OPCODE         (MAX_INTERNAL_OPCODE + 1)

  52 /* Used for non-assigned opcodes */

  54 #define _UNK                        0x6B

  56 /*
  57  * Reserved ASCII characters. Do not use any of these for
  58  * internal opcodes, since they are used to differentiate
  59  * name strings from AML opcodes
```



```
  60  */
  61 #define _ASC                        0x6C
  62 #define _NAM                        0x6C
  63 #define _PFX                        0x6D


  66 /*
  67  * All AML opcodes and the parse-time arguments for each. Used by the AML
  68  * parser  Each list is compressed into a 32-bit number and stored in the
  69  * master opcode table (in psopcode.c).
  70  */
  71 #define ARGP_ACCESSFIELD_OP             ARGP_LIST1 (ARGP_NAMESTRING)
  72 #define ARGP_ACQUIRE_OP                 ARGP_LIST2 (ARGP_SUPERNAME,  ARGP_WORDDA
  73 #define ARGP_ADD_OP                     ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  74 #define ARGP_ALIAS_OP                   ARGP_LIST2 (ARGP_NAMESTRING, ARGP_NAME)
  75 #define ARGP_ARG0                       ARG_NONE
  76 #define ARGP_ARG1                       ARG_NONE
  77 #define ARGP_ARG2                       ARG_NONE
  78 #define ARGP_ARG3                       ARG_NONE
  79 #define ARGP_ARG4                       ARG_NONE
  80 #define ARGP_ARG5                       ARG_NONE
  81 #define ARGP_ARG6                       ARG_NONE
  82 #define ARGP_BANK_FIELD_OP              ARGP_LIST6 (ARGP_PKGLENGTH,  ARGP_NAMEST
  83 #define ARGP_BIT_AND_OP                 ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  84 #define ARGP_BIT_NAND_OP                ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  85 #define ARGP_BIT_NOR_OP                 ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  86 #define ARGP_BIT_NOT_OP                 ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
  87 #define ARGP_BIT_OR_OP                  ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  88 #define ARGP_BIT_XOR_OP                 ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  89 #define ARGP_BREAK_OP                   ARG_NONE
  90 #define ARGP_BREAK_POINT_OP             ARG_NONE
  91 #define ARGP_BUFFER_OP                  ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_TERMAR
  92 #define ARGP_BYTE_OP                    ARGP_LIST1 (ARGP_BYTEDATA)
  93 #define ARGP_BYTELIST_OP                ARGP_LIST1 (ARGP_NAMESTRING)
  94 #define ARGP_CONCAT_OP                  ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  95 #define ARGP_CONCAT_RES_OP              ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
  96 #define ARGP_COND_REF_OF_OP             ARGP_LIST2 (ARGP_SUPERNAME,  ARGP_SUPERN
  97 #define ARGP_CONNECTFIELD_OP            ARGP_LIST1 (ARGP_NAMESTRING)
  98 #define ARGP_CONTINUE_OP                ARG_NONE
  99 #define ARGP_COPY_OP                    ARGP_LIST2 (ARGP_TERMARG,    ARGP_SIMPLE
 100 #define ARGP_CREATE_BIT_FIELD_OP        ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 101 #define ARGP_CREATE_BYTE_FIELD_OP       ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 102 #define ARGP_CREATE_DWORD_FIELD_OP      ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 103 #define ARGP_CREATE_FIELD_OP            ARGP_LIST4 (ARGP_TERMARG,    ARGP_TERMAR
 104 #define ARGP_CREATE_QWORD_FIELD_OP      ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 105 #define ARGP_CREATE_WORD_FIELD_OP       ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 106 #define ARGP_DATA_REGION_OP             ARGP_LIST4 (ARGP_NAME,       ARGP_TERMAR
 107 #define ARGP_DEBUG_OP                   ARG_NONE
 108 #define ARGP_DECREMENT_OP               ARGP_LIST1 (ARGP_SUPERNAME)
 109 #define ARGP_DEREF_OF_OP                ARGP_LIST1 (ARGP_TERMARG)
 110 #define ARGP_DEVICE_OP                  ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_NAME,
 111 #define ARGP_DIVIDE_OP                  ARGP_LIST4 (ARGP_TERMARG,    ARGP_TERMAR
 112 #define ARGP_DWORD_OP                   ARGP_LIST1 (ARGP_DWORDDATA)
 113 #define ARGP_ELSE_OP                    ARGP_LIST2 (ARGP_PKGLENGTH,  ARGP_TERMLI
 114 #define ARGP_EVENT_OP                   ARGP_LIST1 (ARGP_NAME)
 115 #define ARGP_FATAL_OP                   ARGP_LIST3 (ARGP_BYTEDATA,   ARGP_DWORDD
 116 #define ARGP_FIELD_OP                   ARGP_LIST4 (ARGP_PKGLENGTH,  ARGP_NAMEST
 117 #define ARGP_FIND_SET_LEFT_BIT_OP       ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
 118 #define ARGP_FIND_SET_RIGHT_BIT_OP      ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
 119 #define ARGP_FROM_BCD_OP                ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
 120 #define ARGP_IF_OP                      ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_TERMAR
 121 #define ARGP_INCREMENT_OP               ARGP_LIST1 (ARGP_SUPERNAME)
 122 #define ARGP_INDEX_FIELD_OP             ARGP_LIST5 (ARGP_PKGLENGTH,  ARGP_NAMEST
 123 #define ARGP_INDEX_OP                   ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
 124 #define ARGP_LAND_OP                    ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
 125 #define ARGP_LEQUAL_OP                  ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
```

```
126 #define ARGP_LGREATER_OP                ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
127 #define ARGP_LGREATEREQUAL_OP           ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
128 #define ARGP_LLESS_OP                   ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
129 #define ARGP_LLESSEQUAL_OP              ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
130 #define ARGP_LNOT_OP                    ARGP_LIST1 (ARGP_TERMARG)
131 #define ARGP_LNOTEQUAL_OP               ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
132 #define ARGP_LOAD_OP                    ARGP_LIST2 (ARGP_NAMESTRING, ARGP_SUPERN
133 #define ARGP_LOAD_TABLE_OP              ARGP_LIST6 (ARGP_TERMARG,    ARGP_TERMAR
134 #define ARGP_LOCAL0                     ARG_NONE
135 #define ARGP_LOCAL1                     ARG_NONE
136 #define ARGP_LOCAL2                     ARG_NONE
137 #define ARGP_LOCAL3                     ARG_NONE
138 #define ARGP_LOCAL4                     ARG_NONE
139 #define ARGP_LOCAL5                     ARG_NONE
140 #define ARGP_LOCAL6                     ARG_NONE
141 #define ARGP_LOCAL7                     ARG_NONE
142 #define ARGP_LOR_OP                     ARGP_LIST2 (ARGP_TERMARG,    ARGP_TERMAR
143 #define ARGP_MATCH_OP                   ARGP_LIST6 (ARGP_TERMARG,    ARGP_BYTEDA
144 #define ARGP_METHOD_OP                  ARGP_LIST4 (ARGP_PKGLENGTH,  ARGP_NAME,
145 #define ARGP_METHODCALL_OP              ARGP_LIST1 (ARGP_NAMESTRING)
146 #define ARGP_MID_OP                     ARGP_LIST4 (ARGP_TERMARG,    ARGP_TERMAR
147 #define ARGP_MOD_OP                     ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
148 #define ARGP_MULTIPLY_OP                ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
149 #define ARGP_MUTEX_OP                   ARGP_LIST2 (ARGP_NAME,       ARGP_BYTEDA
150 #define ARGP_NAME_OP                    ARGP_LIST2 (ARGP_NAME,       ARGP_DATAOB
151 #define ARGP_NAMEDFIELD_OP              ARGP_LIST1 (ARGP_NAMESTRING)
152 #define ARGP_NAMEPATH_OP                ARGP_LIST1 (ARGP_NAMESTRING)
153 #define ARGP_NOOP_OP                    ARG_NONE
154 #define ARGP_NOTIFY_OP                  ARGP_LIST2 (ARGP_SUPERNAME,  ARGP_TERMAR
155 #define ARGP_ONE_OP                     ARG_NONE
156 #define ARGP_ONES_OP                    ARG_NONE
157 #define ARGP_PACKAGE_OP                 ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_BYTEDA
158 #define ARGP_POWER_RES_OP               ARGP_LIST5 (ARGP_PKGLENGTH,  ARGP_NAME,
159 #define ARGP_PROCESSOR_OP               ARGP_LIST6 (ARGP_PKGLENGTH,  ARGP_NAME,
160 #define ARGP_QWORD_OP                   ARGP_LIST1 (ARGP_QWORDDATA)
161 #define ARGP_REF_OF_OP                  ARGP_LIST1 (ARGP_SUPERNAME)
162 #define ARGP_REGION_OP                  ARGP_LIST4 (ARGP_NAME,       ARGP_BYTEDA
163 #define ARGP_RELEASE_OP                 ARGP_LIST1 (ARGP_SUPERNAME)
164 #define ARGP_RESERVEDFIELD_OP           ARGP_LIST1 (ARGP_NAMESTRING)
165 #define ARGP_RESET_OP                   ARGP_LIST1 (ARGP_SUPERNAME)
166 #define ARGP_RETURN_OP                  ARGP_LIST1 (ARGP_TERMARG)
167 #define ARGP_REVISION_OP                ARG_NONE
168 #define ARGP_SCOPE_OP                   ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_NAME,
169 #define ARGP_SERIALFIELD_OP             ARGP_LIST1 (ARGP_NAMESTRING)
170 #define ARGP_SHIFT_LEFT_OP              ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
171 #define ARGP_SHIFT_RIGHT_OP             ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
172 #define ARGP_SIGNAL_OP                  ARGP_LIST1 (ARGP_SUPERNAME)
173 #define ARGP_SIZE_OF_OP                 ARGP_LIST1 (ARGP_SUPERNAME)
174 #define ARGP_SLEEP_OP                   ARGP_LIST1 (ARGP_TERMARG)
175 #define ARGP_STALL_OP                   ARGP_LIST1 (ARGP_TERMARG)
176 #define ARGP_STATICSTRING_OP            ARGP_LIST1 (ARGP_NAMESTRING)
177 #define ARGP_STORE_OP                   ARGP_LIST2 (ARGP_TERMARG,    ARGP_SUPERN
178 #define ARGP_STRING_OP                  ARGP_LIST1 (ARGP_CHARLIST)
179 #define ARGP_SUBTRACT_OP                ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
180 #define ARGP_THERMAL_ZONE_OP            ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_NAME,
181 #define ARGP_TIMER_OP                   ARG_NONE
182 #define ARGP_TO_BCD_OP                  ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
183 #define ARGP_TO_BUFFER_OP               ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
184 #define ARGP_TO_DEC_STR_OP              ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
185 #define ARGP_TO_HEX_STR_OP              ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
186 #define ARGP_TO_INTEGER_OP              ARGP_LIST2 (ARGP_TERMARG,    ARGP_TARGET
187 #define ARGP_TO_STRING_OP               ARGP_LIST3 (ARGP_TERMARG,    ARGP_TERMAR
188 #define ARGP_TYPE_OP                    ARGP_LIST1 (ARGP_SUPERNAME)
189 #define ARGP_UNLOAD_OP                  ARGP_LIST1 (ARGP_SUPERNAME)
190 #define ARGP_VAR_PACKAGE_OP             ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_TERMAR
191 #define ARGP_WAIT_OP                    ARGP_LIST2 (ARGP_SUPERNAME,  ARGP_TERMAR
```



```
192 #define ARGP_WHILE_OP                   ARGP_LIST3 (ARGP_PKGLENGTH,  ARGP_TERMAR
193 #define ARGP_WORD_OP                    ARGP_LIST1 (ARGP_WORDDATA)
194 #define ARGP_ZERO_OP                    ARG_NONE


197 /*
198  * All AML opcodes and the runtime arguments for each. Used by the AML
199  * interpreter  Each list is compressed into a 32-bit number and stored
200  * in the master opcode table (in psopcode.c).
201  *
202  * (Used by PrepOperands procedure and the ASL Compiler)
203  */
204 #define ARGI_ACCESSFIELD_OP             ARGI_INVALID_OPCODE
205 #define ARGI_ACQUIRE_OP                 ARGI_LIST2 (ARGI_MUTEX,      ARGI_INTEGE
206 #define ARGI_ADD_OP                     ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
207 #define ARGI_ALIAS_OP                   ARGI_INVALID_OPCODE
208 #define ARGI_ARG0                       ARG_NONE
209 #define ARGI_ARG1                       ARG_NONE
210 #define ARGI_ARG2                       ARG_NONE
211 #define ARGI_ARG3                       ARG_NONE
212 #define ARGI_ARG4                       ARG_NONE
213 #define ARGI_ARG5                       ARG_NONE
214 #define ARGI_ARG6                       ARG_NONE
215 #define ARGI_BANK_FIELD_OP              ARGI_INVALID_OPCODE
216 #define ARGI_BIT_AND_OP                 ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
217 #define ARGI_BIT_NAND_OP                ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
218 #define ARGI_BIT_NOR_OP                 ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
219 #define ARGI_BIT_NOT_OP                 ARGI_LIST2 (ARGI_INTEGER,    ARGI_TARGET
220 #define ARGI_BIT_OR_OP                  ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
221 #define ARGI_BIT_XOR_OP                 ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
222 #define ARGI_BREAK_OP                   ARG_NONE
223 #define ARGI_BREAK_POINT_OP             ARG_NONE
224 #define ARGI_BUFFER_OP                  ARGI_LIST1 (ARGI_INTEGER)
225 #define ARGI_BYTE_OP                    ARGI_INVALID_OPCODE
226 #define ARGI_BYTELIST_OP                ARGI_INVALID_OPCODE
227 #define ARGI_CONCAT_OP                  ARGI_LIST3 (ARGI_COMPUTEDATA,ARGI_COMPUT
228 #define ARGI_CONCAT_RES_OP              ARGI_LIST3 (ARGI_BUFFER,     ARGI_BUFFER
229 #define ARGI_COND_REF_OF_OP             ARGI_LIST2 (ARGI_OBJECT_REF, ARGI_TARGET
230 #define ARGI_CONNECTFIELD_OP            ARGI_INVALID_OPCODE
231 #define ARGI_CONTINUE_OP                ARGI_INVALID_OPCODE
232 #define ARGI_COPY_OP                    ARGI_LIST2 (ARGI_ANYTYPE,    ARGI_SIMPLE
233 #define ARGI_CREATE_BIT_FIELD_OP        ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
234 #define ARGI_CREATE_BYTE_FIELD_OP       ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
235 #define ARGI_CREATE_DWORD_FIELD_OP      ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
236 #define ARGI_CREATE_FIELD_OP            ARGI_LIST4 (ARGI_BUFFER,     ARGI_INTEGE
237 #define ARGI_CREATE_QWORD_FIELD_OP      ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
238 #define ARGI_CREATE_WORD_FIELD_OP       ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
239 #define ARGI_DATA_REGION_OP             ARGI_LIST3 (ARGI_STRING,     ARGI_STRING
240 #define ARGI_DEBUG_OP                   ARG_NONE
241 #define ARGI_DECREMENT_OP               ARGI_LIST1 (ARGI_TARGETREF)
242 #define ARGI_DEREF_OF_OP                ARGI_LIST1 (ARGI_REF_OR_STRING)
243 #define ARGI_DEVICE_OP                  ARGI_INVALID_OPCODE
244 #define ARGI_DIVIDE_OP                  ARGI_LIST4 (ARGI_INTEGER,    ARGI_INTEGE
245 #define ARGI_DWORD_OP                   ARGI_INVALID_OPCODE
246 #define ARGI_ELSE_OP                    ARGI_INVALID_OPCODE
247 #define ARGI_EVENT_OP                   ARGI_INVALID_OPCODE
248 #define ARGI_FATAL_OP                   ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
249 #define ARGI_FIELD_OP                   ARGI_INVALID_OPCODE
250 #define ARGI_FIND_SET_LEFT_BIT_OP       ARGI_LIST2 (ARGI_INTEGER,    ARGI_TARGET
251 #define ARGI_FIND_SET_RIGHT_BIT_OP      ARGI_LIST2 (ARGI_INTEGER,    ARGI_TARGET
252 #define ARGI_FROM_BCD_OP                ARGI_LIST2 (ARGI_INTEGER,    ARGI_FIXED_
253 #define ARGI_IF_OP                      ARGI_INVALID_OPCODE
254 #define ARGI_INCREMENT_OP               ARGI_LIST1 (ARGI_TARGETREF)
255 #define ARGI_INDEX_FIELD_OP             ARGI_INVALID_OPCODE
256 #define ARGI_INDEX_OP                   ARGI_LIST3 (ARGI_COMPLEXOBJ, ARGI_INTEGE
257 #define ARGI_LAND_OP                    ARGI_LIST2 (ARGI_INTEGER,    ARGI_INTEGE
```

```
258 #define ARGI_LEQUAL_OP                  ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_COMPUT
259 #define ARGI_LGREATER_OP                ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_COMPUT
260 #define ARGI_LGREATEREQUAL_OP           ARGI_INVALID_OPCODE
261 #define ARGI_LLESS_OP                   ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_COMPUT
262 #define ARGI_LLESSEQUAL_OP              ARGI_INVALID_OPCODE
263 #define ARGI_LNOT_OP                    ARGI_LIST1 (ARGI_INTEGER)
264 #define ARGI_LNOTEQUAL_OP               ARGI_INVALID_OPCODE
265 #define ARGI_LOAD_OP                    ARGI_LIST2 (ARGI_REGION_OR_BUFFER,ARGI_T
266 #define ARGI_LOAD_TABLE_OP              ARGI_LIST6 (ARGI_STRING,     ARGI_STRING
267 #define ARGI_LOCAL0                     ARG_NONE
268 #define ARGI_LOCAL1                     ARG_NONE
269 #define ARGI_LOCAL2                     ARG_NONE
270 #define ARGI_LOCAL3                     ARG_NONE
271 #define ARGI_LOCAL4                     ARG_NONE
272 #define ARGI_LOCAL5                     ARG_NONE
273 #define ARGI_LOCAL6                     ARG_NONE
274 #define ARGI_LOCAL7                     ARG_NONE
275 #define ARGI_LOR_OP                     ARGI_LIST2 (ARGI_INTEGER,    ARGI_INTEGE
276 #define ARGI_MATCH_OP                   ARGI_LIST6 (ARGI_PACKAGE,    ARGI_INTEGE
277 #define ARGI_METHOD_OP                  ARGI_INVALID_OPCODE
278 #define ARGI_METHODCALL_OP              ARGI_INVALID_OPCODE
279 #define ARGI_MID_OP                     ARGI_LIST4 (ARGI_BUFFER_OR_STRING,ARGI_I
280 #define ARGI_MOD_OP                     ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
281 #define ARGI_MULTIPLY_OP                ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
282 #define ARGI_MUTEX_OP                   ARGI_INVALID_OPCODE
283 #define ARGI_NAME_OP                    ARGI_INVALID_OPCODE
284 #define ARGI_NAMEDFIELD_OP              ARGI_INVALID_OPCODE
285 #define ARGI_NAMEPATH_OP                ARGI_INVALID_OPCODE
286 #define ARGI_NOOP_OP                    ARG_NONE
287 #define ARGI_NOTIFY_OP                  ARGI_LIST2 (ARGI_DEVICE_REF, ARGI_INTEGE
288 #define ARGI_ONE_OP                     ARG_NONE
289 #define ARGI_ONES_OP                    ARG_NONE
290 #define ARGI_PACKAGE_OP                 ARGI_LIST1 (ARGI_INTEGER)
291 #define ARGI_POWER_RES_OP               ARGI_INVALID_OPCODE
292 #define ARGI_PROCESSOR_OP               ARGI_INVALID_OPCODE
293 #define ARGI_QWORD_OP                   ARGI_INVALID_OPCODE
294 #define ARGI_REF_OF_OP                  ARGI_LIST1 (ARGI_OBJECT_REF)
295 #define ARGI_REGION_OP                  ARGI_LIST2 (ARGI_INTEGER,    ARGI_INTEGE
296 #define ARGI_RELEASE_OP                 ARGI_LIST1 (ARGI_MUTEX)
297 #define ARGI_RESERVEDFIELD_OP           ARGI_INVALID_OPCODE
298 #define ARGI_RESET_OP                   ARGI_LIST1 (ARGI_EVENT)
299 #define ARGI_RETURN_OP                  ARGI_INVALID_OPCODE
300 #define ARGI_REVISION_OP                ARG_NONE
301 #define ARGI_SCOPE_OP                   ARGI_INVALID_OPCODE
302 #define ARGI_SERIALFIELD_OP             ARGI_INVALID_OPCODE
303 #define ARGI_SHIFT_LEFT_OP              ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
304 #define ARGI_SHIFT_RIGHT_OP             ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
305 #define ARGI_SIGNAL_OP                  ARGI_LIST1 (ARGI_EVENT)
306 #define ARGI_SIZE_OF_OP                 ARGI_LIST1 (ARGI_DATAOBJECT)
307 #define ARGI_SLEEP_OP                   ARGI_LIST1 (ARGI_INTEGER)
308 #define ARGI_STALL_OP                   ARGI_LIST1 (ARGI_INTEGER)
309 #define ARGI_STATICSTRING_OP            ARGI_INVALID_OPCODE
310 #define ARGI_STORE_OP                   ARGI_LIST2 (ARGI_DATAREFOBJ, ARGI_TARGET
311 #define ARGI_STRING_OP                  ARGI_INVALID_OPCODE
312 #define ARGI_SUBTRACT_OP                ARGI_LIST3 (ARGI_INTEGER,    ARGI_INTEGE
313 #define ARGI_THERMAL_ZONE_OP            ARGI_INVALID_OPCODE
314 #define ARGI_TIMER_OP                   ARG_NONE
315 #define ARGI_TO_BCD_OP                  ARGI_LIST2 (ARGI_INTEGER,    ARGI_FIXED_
316 #define ARGI_TO_BUFFER_OP               ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_FIXED_
317 #define ARGI_TO_DEC_STR_OP              ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_FIXED_
318 #define ARGI_TO_HEX_STR_OP              ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_FIXED_
319 #define ARGI_TO_INTEGER_OP              ARGI_LIST2 (ARGI_COMPUTEDATA,ARGI_FIXED_
320 #define ARGI_TO_STRING_OP               ARGI_LIST3 (ARGI_BUFFER,     ARGI_INTEGE
321 #define ARGI_TYPE_OP                    ARGI_LIST1 (ARGI_ANYTYPE)
322 #define ARGI_UNLOAD_OP                  ARGI_LIST1 (ARGI_DDBHANDLE)
323 #define ARGI_VAR_PACKAGE_OP             ARGI_LIST1 (ARGI_INTEGER)
```



```
324 #define ARGI_WAIT_OP                    ARGI_LIST2 (ARGI_EVENT,      ARGI_INTEGE
325 #define ARGI_WHILE_OP                   ARGI_INVALID_OPCODE
326 #define ARGI_WORD_OP                    ARGI_INVALID_OPCODE
327 #define ARGI_ZERO_OP                    ARG_NONE

329 #endif /* __ACOPCODE_H__ */
```

```
**********************************************************
   17260 Thu Dec 26 13:49:52 2013
new/usr/src/common/acpica/include/acoutput.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acoutput.h -- debug output
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACOUTPUT_H__
  45 #define __ACOUTPUT_H__

  47 /*
  48  * Debug levels and component IDs. These are used to control the
  49  * granularity of the output of the ACPI_DEBUG_PRINT macro -- on a
  50  * per-component basis and a per-exception-type basis.
  51  */

  53 /* Component IDs are used in the global "DebugLayer" */

  55 #define ACPI_UTILITIES              0x00000001
  56 #define ACPI_HARDWARE               0x00000002
  57 #define ACPI_EVENTS                 0x00000004
  58 #define ACPI_TABLES                 0x00000008
  59 #define ACPI_NAMESPACE              0x00000010
  60 #define ACPI_PARSER                 0x00000020
```



```
  61 #define ACPI_DISPATCHER             0x00000040
  62 #define ACPI_EXECUTER               0x00000080
  63 #define ACPI_RESOURCES              0x00000100
  64 #define ACPI_CA_DEBUGGER            0x00000200
  65 #define ACPI_OS_SERVICES            0x00000400
  66 #define ACPI_CA_DISASSEMBLER        0x00000800

  68 /* Component IDs for ACPI tools and utilities */

  70 #define ACPI_COMPILER               0x00001000
  71 #define ACPI_TOOLS                  0x00002000
  72 #define ACPI_EXAMPLE                0x00004000
  73 #define ACPI_DRIVER                 0x00008000
  74 #define DT_COMPILER                 0x00010000
  75 #define ASL_PREPROCESSOR            0x00020000

  77 #define ACPI_ALL_COMPONENTS         0x0001FFFF
  78 #define ACPI_COMPONENT_DEFAULT      (ACPI_ALL_COMPONENTS)

  80 /* Component IDs reserved for ACPI drivers */

  82 #define ACPI_ALL_DRIVERS            0xFFFF0000


  85 /*
  86  * Raw debug output levels, do not use these in the ACPI_DEBUG_PRINT macros
  87  */
  88 #define ACPI_LV_INIT                0x00000001
  89 #define ACPI_LV_DEBUG_OBJECT        0x00000002
  90 #define ACPI_LV_INFO                0x00000004
  91 #define ACPI_LV_REPAIR              0x00000008
  92 #define ACPI_LV_ALL_EXCEPTIONS      0x0000000F

  94 /* Trace verbosity level 1 [Standard Trace Level] */

  96 #define ACPI_LV_INIT_NAMES          0x00000020
  97 #define ACPI_LV_PARSE               0x00000040
  98 #define ACPI_LV_LOAD                0x00000080
  99 #define ACPI_LV_DISPATCH            0x00000100
 100 #define ACPI_LV_EXEC                0x00000200
 101 #define ACPI_LV_NAMES               0x00000400
 102 #define ACPI_LV_OPREGION            0x00000800
 103 #define ACPI_LV_BFIELD              0x00001000
 104 #define ACPI_LV_TABLES              0x00002000
 105 #define ACPI_LV_VALUES              0x00004000
 106 #define ACPI_LV_OBJECTS             0x00008000
 107 #define ACPI_LV_RESOURCES           0x00010000
 108 #define ACPI_LV_USER_REQUESTS       0x00020000
 109 #define ACPI_LV_PACKAGE             0x00040000
 110 #define ACPI_LV_VERBOSITY1          0x0007FF40 | ACPI_LV_ALL_EXCEPTIONS

 112 /* Trace verbosity level 2 [Function tracing and memory allocation] */

 114 #define ACPI_LV_ALLOCATIONS         0x00100000
 115 #define ACPI_LV_FUNCTIONS           0x00200000
 116 #define ACPI_LV_OPTIMIZATIONS       0x00400000
 117 #define ACPI_LV_VERBOSITY2          0x00700000 | ACPI_LV_VERBOSITY1
 118 #define ACPI_LV_ALL                 ACPI_LV_VERBOSITY2

 120 /* Trace verbosity level 3 [Threading, I/O, and Interrupts] */

 122 #define ACPI_LV_MUTEX               0x01000000
 123 #define ACPI_LV_THREADS             0x02000000
 124 #define ACPI_LV_IO                  0x04000000
 125 #define ACPI_LV_INTERRUPTS          0x08000000
 126 #define ACPI_LV_VERBOSITY3          0x0F000000 | ACPI_LV_VERBOSITY2
```

```
/* Exceptionally verbose output -- also used in the global "DebugLevel"  */

#define ACPI_LV_AML_DISASSEMBLE     0x10000000
#define ACPI_LV_VERBOSE_INFO        0x20000000
#define ACPI_LV_FULL_TABLES         0x40000000
#define ACPI_LV_EVENTS              0x80000000
#define ACPI_LV_VERBOSE             0xF0000000


/*
 * Debug level macros that are used in the DEBUG_PRINT macros
 */
#define ACPI_DEBUG_LEVEL(dl)        (UINT32) dl,ACPI_DEBUG_PARAMETERS

/*
 * Exception level -- used in the global "DebugLevel"
 *
 * Note: For errors, use the ACPI_ERROR or ACPI_EXCEPTION interfaces.
 * For warnings, use ACPI_WARNING.
 */
#define ACPI_DB_INIT                ACPI_DEBUG_LEVEL (ACPI_LV_INIT)
#define ACPI_DB_DEBUG_OBJECT        ACPI_DEBUG_LEVEL (ACPI_LV_DEBUG_OBJECT)
#define ACPI_DB_INFO                ACPI_DEBUG_LEVEL (ACPI_LV_INFO)
#define ACPI_DB_REPAIR              ACPI_DEBUG_LEVEL (ACPI_LV_REPAIR)
#define ACPI_DB_ALL_EXCEPTIONS      ACPI_DEBUG_LEVEL (ACPI_LV_ALL_EXCEPTIONS)

/* Trace level -- also used in the global "DebugLevel" */

#define ACPI_DB_INIT_NAMES          ACPI_DEBUG_LEVEL (ACPI_LV_INIT_NAMES)
#define ACPI_DB_THREADS             ACPI_DEBUG_LEVEL (ACPI_LV_THREADS)
#define ACPI_DB_PARSE               ACPI_DEBUG_LEVEL (ACPI_LV_PARSE)
#define ACPI_DB_DISPATCH            ACPI_DEBUG_LEVEL (ACPI_LV_DISPATCH)
#define ACPI_DB_LOAD                ACPI_DEBUG_LEVEL (ACPI_LV_LOAD)
#define ACPI_DB_EXEC                ACPI_DEBUG_LEVEL (ACPI_LV_EXEC)
#define ACPI_DB_NAMES               ACPI_DEBUG_LEVEL (ACPI_LV_NAMES)
#define ACPI_DB_OPREGION            ACPI_DEBUG_LEVEL (ACPI_LV_OPREGION)
#define ACPI_DB_BFIELD              ACPI_DEBUG_LEVEL (ACPI_LV_BFIELD)
#define ACPI_DB_TABLES              ACPI_DEBUG_LEVEL (ACPI_LV_TABLES)
#define ACPI_DB_FUNCTIONS           ACPI_DEBUG_LEVEL (ACPI_LV_FUNCTIONS)
#define ACPI_DB_OPTIMIZATIONS       ACPI_DEBUG_LEVEL (ACPI_LV_OPTIMIZATIONS)
#define ACPI_DB_VALUES              ACPI_DEBUG_LEVEL (ACPI_LV_VALUES)
#define ACPI_DB_OBJECTS             ACPI_DEBUG_LEVEL (ACPI_LV_OBJECTS)
#define ACPI_DB_ALLOCATIONS         ACPI_DEBUG_LEVEL (ACPI_LV_ALLOCATIONS)
#define ACPI_DB_RESOURCES           ACPI_DEBUG_LEVEL (ACPI_LV_RESOURCES)
#define ACPI_DB_IO                  ACPI_DEBUG_LEVEL (ACPI_LV_IO)
#define ACPI_DB_INTERRUPTS          ACPI_DEBUG_LEVEL (ACPI_LV_INTERRUPTS)
#define ACPI_DB_USER_REQUESTS       ACPI_DEBUG_LEVEL (ACPI_LV_USER_REQUESTS)
#define ACPI_DB_PACKAGE             ACPI_DEBUG_LEVEL (ACPI_LV_PACKAGE)
#define ACPI_DB_MUTEX               ACPI_DEBUG_LEVEL (ACPI_LV_MUTEX)
#define ACPI_DB_EVENTS              ACPI_DEBUG_LEVEL (ACPI_LV_EVENTS)

#define ACPI_DB_ALL                 ACPI_DEBUG_LEVEL (ACPI_LV_ALL)

/* Defaults for DebugLevel, debug and normal */

#define ACPI_DEBUG_DEFAULT          (ACPI_LV_INIT | ACPI_LV_DEBUG_OBJECT | ACPI_
#define ACPI_NORMAL_DEFAULT         (ACPI_LV_INIT | ACPI_LV_DEBUG_OBJECT | ACPI_
#define ACPI_DEBUG_ALL              (ACPI_LV_AML_DISASSEMBLE | ACPI_LV_ALL_EXCEP


#if defined (ACPI_DEBUG_OUTPUT) || !defined (ACPI_NO_ERROR_MESSAGES)
/*
 * The module name is used primarily for error and debug messages.
 * The __FILE__ macro is not very useful for this, because it
 * usually includes the entire pathname to the module making the
```



```
 * debug output difficult to read.
 */
#define ACPI_MODULE_NAME(Name)          static const char ACPI_UNUSED_VAR _AcpiM
#else
/*
 * For the no-debug and no-error-msg cases, we must at least define
 * a null module name.
 */
#define ACPI_MODULE_NAME(Name)
#define _AcpiModuleName ""
#endif

/*
 * Ascii error messages can be configured out
 */
#ifndef ACPI_NO_ERROR_MESSAGES
#define AE_INFO                         _AcpiModuleName, __LINE__

/*
 * Error reporting. Callers module and line number are inserted by AE_INFO,
 * the plist contains a set of parens to allow variable-length lists.
 * These macros are used for both the debug and non-debug versions of the code.
 */
#define ACPI_INFO(plist)                AcpiInfo plist
#define ACPI_WARNING(plist)             AcpiWarning plist
#define ACPI_EXCEPTION(plist)           AcpiException plist
#define ACPI_ERROR(plist)               AcpiError plist
#define ACPI_BIOS_WARNING(plist)        AcpiBiosWarning plist
#define ACPI_BIOS_ERROR(plist)          AcpiBiosError plist
#define ACPI_DEBUG_OBJECT(obj,l,i)      AcpiExDoDebugObject(obj,l,i)

#else

/* No error messages */

#define ACPI_INFO(plist)
#define ACPI_WARNING(plist)
#define ACPI_EXCEPTION(plist)
#define ACPI_ERROR(plist)
#define ACPI_BIOS_WARNING(plist)
#define ACPI_BIOS_ERROR(plist)
#define ACPI_DEBUG_OBJECT(obj,l,i)

#endif /* ACPI_NO_ERROR_MESSAGES */


/*
 * Debug macros that are conditionally compiled
 */
#ifdef ACPI_DEBUG_OUTPUT

/*
 * If ACPI_GET_FUNCTION_NAME was not defined in the compiler-dependent header,
 * define it now. This is the case where there the compiler does not support
 * a __FUNCTION__ macro or equivalent.
 */
#ifndef ACPI_GET_FUNCTION_NAME
#define ACPI_GET_FUNCTION_NAME          _AcpiFunctionName

/*
 * The Name parameter should be the procedure name as a quoted string.
 * The function name is also used by the function exit macros below.
 * Note: (const char) is used to be compatible with the debug interfaces
 * and macros such as __FUNCTION__.
 */
#define ACPI_FUNCTION_NAME(Name)        static const char _AcpiFunctionName[] =
```

```
#else
/* Compiler supports __FUNCTION__ (or equivalent) -- Ignore this macro */

#define ACPI_FUNCTION_NAME(Name)
#endif /* ACPI_GET_FUNCTION_NAME */

/*
 * Common parameters used for debug output functions:
 * line number, function name, module(file) name, component ID
 */
#define ACPI_DEBUG_PARAMETERS \
    __LINE__, ACPI_GET_FUNCTION_NAME, _AcpiModuleName, _COMPONENT

/* Check if debug output is currently dynamically enabled */

#define ACPI_IS_DEBUG_ENABLED(Level, Component) \
    ((Level & AcpiDbgLevel) && (Component & AcpiDbgLayer))

/*
 * Master debug print macros
 * Print message if and only if:
 *    1) Debug print for the current component is enabled
 *    2) Debug error level or trace level for the print statement is enabled
 *
 * November 2012: Moved the runtime check for whether to actually emit the
 * debug message outside of the print function itself. This improves overall
 * performance at a relatively small code cost. Implementation involves the
 * use of variadic macros supported by C99.
 *
 * Note: the ACPI_DO_WHILE0 macro is used to prevent some compilers from
 * complaining about these constructs. On other compilers the do...while
 * adds some extra code, so this feature is optional.
 */
#ifdef ACPI_USE_DO_WHILE_0
#define ACPI_DO_WHILE0(a)               do a while(0)
#else
#define ACPI_DO_WHILE0(a)               a
#endif

/* DEBUG_PRINT functions */

#define ACPI_DEBUG_PRINT(plist)         ACPI_ACTUAL_DEBUG plist
#define ACPI_DEBUG_PRINT_RAW(plist)     ACPI_ACTUAL_DEBUG_RAW plist

/* Helper macros for DEBUG_PRINT */

#define ACPI_DO_DEBUG_PRINT(Function, Level, Line, Filename, Modulename, Compone
    ACPI_DO_WHILE0 ({ \
        if (ACPI_IS_DEBUG_ENABLED (Level, Component)) \
        { \
            Function (Level, Line, Filename, Modulename, Component, __VA_ARGS__)
        } \
    })

#define ACPI_ACTUAL_DEBUG(Level, Line, Filename, Modulename, Component, ...) \
    ACPI_DO_DEBUG_PRINT (AcpiDebugPrint, Level, Line, \
        Filename, Modulename, Component, __VA_ARGS__)

#define ACPI_ACTUAL_DEBUG_RAW(Level, Line, Filename, Modulename, Component, ...)
    ACPI_DO_DEBUG_PRINT (AcpiDebugPrintRaw, Level, Line, \
        Filename, Modulename, Component, __VA_ARGS__)


/*
 * Function entry tracing
```



```
 *
 * The name of the function is emitted as a local variable that is
 * intended to be used by both the entry trace and the exit trace.
 */

/* Helper macro */

#define ACPI_TRACE_ENTRY(Name, Function, Type, Param) \
    ACPI_FUNCTION_NAME (Name) \
    Function (ACPI_DEBUG_PARAMETERS, (Type) (Param))

/* The actual entry trace macros */

#define ACPI_FUNCTION_TRACE(Name) \
    ACPI_FUNCTION_NAME(Name) \
    AcpiUtTrace (ACPI_DEBUG_PARAMETERS)

#define ACPI_FUNCTION_TRACE_PTR(Name, Pointer) \
    ACPI_TRACE_ENTRY (Name, AcpiUtTracePtr, void *, Pointer)

#define ACPI_FUNCTION_TRACE_U32(Name, Value) \
    ACPI_TRACE_ENTRY (Name, AcpiUtTraceU32, UINT32, Value)

#define ACPI_FUNCTION_TRACE_STR(Name, String) \
    ACPI_TRACE_ENTRY (Name, AcpiUtTraceStr, char *, String)

#define ACPI_FUNCTION_ENTRY() \
    AcpiUtTrackStackPtr()


/*
 * Function exit tracing
 *
 * These macros include a return statement. This is usually considered
 * bad form, but having a separate exit macro before the actual return
 * is very ugly and difficult to maintain.
 *
 * One of the FUNCTION_TRACE macros above must be used in conjunction
 * with these macros so that "_AcpiFunctionName" is defined.
 *
 * There are two versions of most of the return macros. The default version is
 * safer, since it avoids side-effects by guaranteeing that the argument will
 * not be evaluated twice.
 *
 * A less-safe version of the macros is provided for optional use if the
 * compiler uses excessive CPU stack (for example, this may happen in the
 * debug case if code optimzation is disabled.)
 */

/* Exit trace helper macro */

#ifndef ACPI_SIMPLE_RETURN_MACROS

#define ACPI_TRACE_EXIT(Function, Type, Param) \
    ACPI_DO_WHILE0 ({ \
        register Type _Param = (Type) (Param); \
        Function (ACPI_DEBUG_PARAMETERS, _Param); \
        return (_Param); \
    })

#else /* Use original less-safe macros */

#define ACPI_TRACE_EXIT(Function, Type, Param) \
    ACPI_DO_WHILE0 ({ \
        Function (ACPI_DEBUG_PARAMETERS, (Type) (Param)); \
        return (Param); \
```

```
391     })

393 #endif /* ACPI_SIMPLE_RETURN_MACROS */

395 /* The actual exit macros */

397 #define return_VOID \
398     ACPI_DO_WHILE0 ({ \
399         AcpiUtExit (ACPI_DEBUG_PARAMETERS); \
400         return; \
401     })

403 #define return_ACPI_STATUS(Status) \
404     ACPI_TRACE_EXIT (AcpiUtStatusExit, ACPI_STATUS, Status)

406 #define return_PTR(Pointer) \
407     ACPI_TRACE_EXIT (AcpiUtPtrExit, void *, Pointer)

409 #define return_VALUE(Value) \
410     ACPI_TRACE_EXIT (AcpiUtValueExit, UINT64, Value)

412 #define return_UINT32(Value) \
413     ACPI_TRACE_EXIT (AcpiUtValueExit, UINT32, Value)

415 #define return_UINT8(Value) \
416     ACPI_TRACE_EXIT (AcpiUtValueExit, UINT8, Value)

418 /* Conditional execution */

420 #define ACPI_DEBUG_EXEC(a)              a
421 #define ACPI_DEBUG_ONLY_MEMBERS(a)      a;
422 #define _VERBOSE_STRUCTURES


425 /* Various object display routines for debug */

427 #define ACPI_DUMP_STACK_ENTRY(a)        AcpiExDumpOperand((a), 0)
428 #define ACPI_DUMP_OPERANDS(a, b ,c)     AcpiExDumpOperands(a, b, c)
429 #define ACPI_DUMP_ENTRY(a, b)           AcpiNsDumpEntry (a, b)
430 #define ACPI_DUMP_PATHNAME(a, b, c, d)  AcpiNsDumpPathname(a, b, c, d)
431 #define ACPI_DUMP_BUFFER(a, b)          AcpiUtDebugDumpBuffer((UINT8 *) a, b, DB

433 #else /* ACPI_DEBUG_OUTPUT */
434 /*
435  * This is the non-debug case -- make everything go away,
436  * leaving no executable debug code!
437  */
438 #define ACPI_DEBUG_PRINT(pl)
439 #define ACPI_DEBUG_PRINT_RAW(pl)
440 #define ACPI_DEBUG_EXEC(a)
441 #define ACPI_DEBUG_ONLY_MEMBERS(a)
442 #define ACPI_FUNCTION_NAME(a)
443 #define ACPI_FUNCTION_TRACE(a)
444 #define ACPI_FUNCTION_TRACE_PTR(a, b)
445 #define ACPI_FUNCTION_TRACE_U32(a, b)
446 #define ACPI_FUNCTION_TRACE_STR(a, b)
447 #define ACPI_FUNCTION_ENTRY()
448 #define ACPI_DUMP_STACK_ENTRY(a)
449 #define ACPI_DUMP_OPERANDS(a, b, c)
450 #define ACPI_DUMP_ENTRY(a, b)
451 #define ACPI_DUMP_PATHNAME(a, b, c, d)
452 #define ACPI_DUMP_BUFFER(a, b)
453 #define ACPI_IS_DEBUG_ENABLED(Level, Component) 0

455 /* Return macros must have a return statement at the minimum */
```



```
457 #define return_VOID                     return
458 #define return_ACPI_STATUS(s)           return(s)
459 #define return_PTR(s)                   return(s)
460 #define return_VALUE(s)                 return(s)
461 #define return_UINT8(s)                 return(s)
462 #define return_UINT32(s)                return(s)

464 #endif /* ACPI_DEBUG_OUTPUT */


467 #endif /* __ACOUTPUT_H__ */
```

```
**********************************************************
    8630 Thu Dec 26 13:49:52 2013
new/usr/src/common/acpica/include/acparser.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acparser.h - AML Parser subcomponent prototypes and defines
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ACPARSER_H__
#define __ACPARSER_H__


#define OP_HAS_RETURN_VALUE             1

/* Variable number of arguments. This field must be 32 bits */

#define ACPI_VAR_ARGS                   ACPI_UINT32_MAX


#define ACPI_PARSE_DELETE_TREE          0x0001
#define ACPI_PARSE_NO_TREE_DELETE       0x0000
#define ACPI_PARSE_TREE_MASK            0x0001
```



```
#define ACPI_PARSE_LOAD_PASS1           0x0010
#define ACPI_PARSE_LOAD_PASS2           0x0020
#define ACPI_PARSE_EXECUTE              0x0030
#define ACPI_PARSE_MODE_MASK            0x0030

#define ACPI_PARSE_DEFERRED_OP          0x0100
#define ACPI_PARSE_DISASSEMBLE          0x0200

#define ACPI_PARSE_MODULE_LEVEL         0x0400

/******************************************************************************
 *
 * Parser interfaces
 *
 *****************************************************************************/


/*
 * psxface - Parser external interfaces
 */
ACPI_STATUS
AcpiPsExecuteMethod (
    ACPI_EVALUATE_INFO      *Info);


/*
 * psargs - Parse AML opcode arguments
 */
UINT8 *
AcpiPsGetNextPackageEnd (
    ACPI_PARSE_STATE        *ParserState);

char *
AcpiPsGetNextNamestring (
    ACPI_PARSE_STATE        *ParserState);

void
AcpiPsGetNextSimpleArg (
    ACPI_PARSE_STATE        *ParserState,
    UINT32                  ArgType,
    ACPI_PARSE_OBJECT       *Arg);

ACPI_STATUS
AcpiPsGetNextNamepath (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_STATE        *ParserState,
    ACPI_PARSE_OBJECT       *Arg,
    BOOLEAN                 MethodCall);

ACPI_STATUS
AcpiPsGetNextArg (
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_STATE        *ParserState,
    UINT32                  ArgType,
    ACPI_PARSE_OBJECT       **ReturnArg);


/*
 * psfind
 */
ACPI_PARSE_OBJECT *
AcpiPsFindName (
    ACPI_PARSE_OBJECT       *Scope,
    UINT32                  Name,
    UINT32                  Opcode);
```

```
126 ACPI_PARSE_OBJECT*
127 AcpiPsGetParent (
128     ACPI_PARSE_OBJECT       *Op);


131 /*
132  * psobject - support for parse object processing
133  * psopcode - AML Opcode information
134  */
135 ACPI_STATUS
136 AcpiPsBuildNamedOp (
137     ACPI_WALK_STATE         *WalkState,
138     UINT8                   *AmlOpStart,
139     ACPI_PARSE_OBJECT       *UnnamedOp,
140     ACPI_PARSE_OBJECT       **Op);

142 ACPI_STATUS
143 AcpiPsCreateOp (
144     ACPI_WALK_STATE         *WalkState,
145     UINT8                   *AmlOpStart,
146     ACPI_PARSE_OBJECT       **NewOp);

148 ACPI_STATUS
149 AcpiPsCompleteOp (
150     ACPI_WALK_STATE         *WalkState,
151     ACPI_PARSE_OBJECT       **Op,
152     ACPI_STATUS             Status);

154 ACPI_STATUS
155 AcpiPsCompleteFinalOp (
156     ACPI_WALK_STATE         *WalkState,
157     ACPI_PARSE_OBJECT       *Op,
158     ACPI_STATUS             Status);


161 /*
162  * psopinfo - AML Opcode information
163  */
164 const ACPI_OPCODE_INFO *
165 AcpiPsGetOpcodeInfo (
166     UINT16                  Opcode);

168 char *
169 AcpiPsGetOpcodeName (
170     UINT16                  Opcode);

172 UINT8
173 AcpiPsGetArgumentCount (
174     UINT32                  OpType);


177 /*
178  * psparse - top level parsing routines
179  */
180 ACPI_STATUS
181 AcpiPsParseAml (
182     ACPI_WALK_STATE         *WalkState);

184 UINT32
185 AcpiPsGetOpcodeSize (
186     UINT32                  Opcode);

188 UINT16
189 AcpiPsPeekOpcode (
190     ACPI_PARSE_STATE        *state);
```



```
191 ACPI_STATUS
192 AcpiPsCompleteThisOp (
193     ACPI_WALK_STATE         *WalkState,
194     ACPI_PARSE_OBJECT       *Op);

196 ACPI_STATUS
197 AcpiPsNextParseState (
198     ACPI_WALK_STATE         *WalkState,
199     ACPI_PARSE_OBJECT       *Op,
200     ACPI_STATUS             CallbackStatus);


203 /*
204  * psloop - main parse loop
205  */
206 ACPI_STATUS
207 AcpiPsParseLoop (
208     ACPI_WALK_STATE         *WalkState);


211 /*
212  * psscope - Scope stack management routines
213  */
214 ACPI_STATUS
215 AcpiPsInitScope (
216     ACPI_PARSE_STATE        *ParserState,
217     ACPI_PARSE_OBJECT       *Root);

219 ACPI_PARSE_OBJECT *
220 AcpiPsGetParentScope (
221     ACPI_PARSE_STATE        *state);

223 BOOLEAN
224 AcpiPsHasCompletedScope (
225     ACPI_PARSE_STATE        *ParserState);

227 void
228 AcpiPsPopScope (
229     ACPI_PARSE_STATE        *ParserState,
230     ACPI_PARSE_OBJECT       **Op,
231     UINT32                  *ArgList,
232     UINT32                  *ArgCount);

234 ACPI_STATUS
235 AcpiPsPushScope (
236     ACPI_PARSE_STATE        *ParserState,
237     ACPI_PARSE_OBJECT       *Op,
238     UINT32                  RemainingArgs,
239     UINT32                  ArgCount);

241 void
242 AcpiPsCleanupScope (
243     ACPI_PARSE_STATE        *state);


246 /*
247  * pstree - parse tree manipulation routines
248  */
249 void
250 AcpiPsAppendArg(
251     ACPI_PARSE_OBJECT       *op,
252     ACPI_PARSE_OBJECT       *arg);

254 ACPI_PARSE_OBJECT*
255 AcpiPsFind (
256     ACPI_PARSE_OBJECT       *Scope,
```

```
257     char                    *Path,
258     UINT16                  Opcode,
259     UINT32                  Create);

261 ACPI_PARSE_OBJECT *
262 AcpiPsGetArg(
263     ACPI_PARSE_OBJECT       *op,
264     UINT32                   argn);

266 ACPI_PARSE_OBJECT *
267 AcpiPsGetDepthNext (
268     ACPI_PARSE_OBJECT       *Origin,
269     ACPI_PARSE_OBJECT       *Op);


272 /*
273  * pswalk - parse tree walk routines
274  */
275 ACPI_STATUS
276 AcpiPsWalkParsedAml (
277     ACPI_PARSE_OBJECT       *StartOp,
278     ACPI_PARSE_OBJECT       *EndOp,
279     ACPI_OPERAND_OBJECT     *MthDesc,
280     ACPI_NAMESPACE_NODE     *StartNode,
281     ACPI_OPERAND_OBJECT     **Params,
282     ACPI_OPERAND_OBJECT     **CallerReturnDesc,
283     ACPI_OWNER_ID           OwnerId,
284     ACPI_PARSE_DOWNWARDS    DescendingCallback,
285     ACPI_PARSE_UPWARDS      AscendingCallback);

287 ACPI_STATUS
288 AcpiPsGetNextWalkOp (
289     ACPI_WALK_STATE         *WalkState,
290     ACPI_PARSE_OBJECT       *Op,
291     ACPI_PARSE_UPWARDS      AscendingCallback);

293 ACPI_STATUS
294 AcpiPsDeleteCompletedOp (
295     ACPI_WALK_STATE         *WalkState);

297 void
298 AcpiPsDeleteParseTree (
299     ACPI_PARSE_OBJECT       *root);


302 /*
303  * psutils - parser utilities
304  */
305 ACPI_PARSE_OBJECT *
306 AcpiPsCreateScopeOp (
307     void);

309 void
310 AcpiPsInitOp (
311     ACPI_PARSE_OBJECT       *op,
312     UINT16                  opcode);

314 ACPI_PARSE_OBJECT *
315 AcpiPsAllocOp (
316     UINT16                  opcode);

318 void
319 AcpiPsFreeOp (
320     ACPI_PARSE_OBJECT       *Op);

322 BOOLEAN
```



```
323 AcpiPsIsLeadingChar (
324     UINT32                  c);

297 BOOLEAN
298 AcpiPsIsPrefixChar (
299     UINT32                  c);

326 UINT32
327 AcpiPsGetName(
328     ACPI_PARSE_OBJECT       *op);

330 void
331 AcpiPsSetName(
332     ACPI_PARSE_OBJECT       *op,
333     UINT32                  name);


336 /*
337  * psdump - display parser tree
338  */
339 UINT32
340 AcpiPsSprintPath (
341     char                    *BufferStart,
342     UINT32                  BufferSize,
343     ACPI_PARSE_OBJECT       *Op);

345 UINT32
346 AcpiPsSprintOp (
347     char                    *BufferStart,
348     UINT32                  BufferSize,
349     ACPI_PARSE_OBJECT       *Op);

351 void
352 AcpiPsShow (
353     ACPI_PARSE_OBJECT       *op);


356 #endif /* __ACPARSER_H__ */
```

```
**********************************************************
    3105 Thu Dec 26 13:49:53 2013
new/usr/src/common/acpica/include/acpi.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acpi.h - Master public include file used to interface to ACPICA
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACPI_H__
#define __ACPI_H__

/*
 * Public include files for use by code that will interface to ACPICA.
 *
 * Information includes the ACPICA data types, names, exceptions, and
 * external interface prototypes. Also included are the definitions for
 * all ACPI tables (FADT, MADT, etc.)
 *
 * Note: The order of these include files is important.
 */
#include "platform/acenv.h"     /* Environment-specific items */
#include "acnames.h"            /* Common ACPI names and strings */
#include "actypes.h"            /* ACPICA data types and structures */
#include "acexcep.h"            /* ACPICA exceptions */
```



```
#include "actbl.h"              /* ACPI table definitions */
#include "acoutput.h"           /* Error output and Debug macros */
#include "acrestyp.h"           /* Resource Descriptor structs */
#include "acpiosxf.h"           /* OSL interfaces (ACPICA-to-OS) */
#include "acpixf.h"             /* ACPI core subsystem external interfaces */

#endif /* __ACPI_H__ */
```

```
**********************************************************
   13374 Thu Dec 26 13:49:53 2013
new/usr/src/common/acpica/include/acpiosxf.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acpiosxf.h - All interfaces to the OS Services Layer (OSL). These
 *                    interfaces must be implemented by OSL to interface the
 *                    ACPI components to the host operating system.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACPIOSXF_H__
#define __ACPIOSXF_H__

#include "platform/acenv.h"
#include "actypes.h"


/* Types for AcpiOsExecute */

typedef enum
{
    OSL_GLOBAL_LOCK_HANDLER,
    OSL_NOTIFY_HANDLER,
    OSL_GPE_HANDLER,
```



```
    OSL_DEBUGGER_THREAD,
    OSL_EC_POLL_HANDLER,
    OSL_EC_BURST_HANDLER

} ACPI_EXECUTE_TYPE;

#define ACPI_NO_UNIT_LIMIT          ((UINT32) -1)
#define ACPI_MUTEX_SEM              1


/* Functions for AcpiOsSignal */

#define ACPI_SIGNAL_FATAL           0
#define ACPI_SIGNAL_BREAKPOINT      1

typedef struct acpi_signal_fatal_info
{
    UINT32                  Type;
    UINT32                  Code;
    UINT32                  Argument;

} ACPI_SIGNAL_FATAL_INFO;


/*
 * OSL Initialization and shutdown primitives
 */
#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsInitialize
ACPI_STATUS
AcpiOsInitialize (
    void);
#endif

#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsTerminate
ACPI_STATUS
AcpiOsTerminate (
    void);
#endif


/*
 * ACPI Table interfaces
 */
#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetRootPointer
ACPI_PHYSICAL_ADDRESS
AcpiOsGetRootPointer (
    void);
#endif

#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPredefinedOverride
ACPI_STATUS
AcpiOsPredefinedOverride (
    const ACPI_PREDEFINED_NAMES *InitVal,
    ACPI_STRING                 *NewVal);
#endif

#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsTableOverride
ACPI_STATUS
AcpiOsTableOverride (
    ACPI_TABLE_HEADER       *ExistingTable,
    ACPI_TABLE_HEADER       **NewTable);
#endif

#ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPhysicalTableOverride
ACPI_STATUS
AcpiOsPhysicalTableOverride (
```

```
126     ACPI_TABLE_HEADER       *ExistingTable,
127     ACPI_PHYSICAL_ADDRESS   *NewAddress,
128     UINT32                  *NewTableLength);
129 #endif


132 /*
133  * Spinlock primitives
134  */
135 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCreateLock
136 ACPI_STATUS
137 AcpiOsCreateLock (
138     ACPI_SPINLOCK           *OutHandle);
139 #endif

141 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsDeleteLock
142 void
143 AcpiOsDeleteLock (
144     ACPI_SPINLOCK           Handle);
145 #endif

147 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAcquireLock
148 ACPI_CPU_FLAGS
149 AcpiOsAcquireLock (
150     ACPI_SPINLOCK           Handle);
151 #endif

153 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReleaseLock
154 void
155 AcpiOsReleaseLock (
156     ACPI_SPINLOCK           Handle,
157     ACPI_CPU_FLAGS          Flags);
158 #endif


161 /*
162  * Semaphore primitives
163  */
164 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCreateSemaphore
165 ACPI_STATUS
166 AcpiOsCreateSemaphore (
167     UINT32                  MaxUnits,
168     UINT32                  InitialUnits,
169     ACPI_SEMAPHORE          *OutHandle);
170 #endif

172 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsDeleteSemaphore
173 ACPI_STATUS
174 AcpiOsDeleteSemaphore (
175     ACPI_SEMAPHORE          Handle);
176 #endif

178 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWaitSemaphore
179 ACPI_STATUS
180 AcpiOsWaitSemaphore (
181     ACPI_SEMAPHORE          Handle,
182     UINT32                  Units,
183     UINT16                  Timeout);
184 #endif

186 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsSignalSemaphore
187 ACPI_STATUS
188 AcpiOsSignalSemaphore (
189     ACPI_SEMAPHORE          Handle,
190     UINT32                  Units);
191 #endif
```



```


194 /*
195  * Mutex primitives. May be configured to use semaphores instead via
196  * ACPI_MUTEX_TYPE (see platform/acenv.h)
197  */
198 #if (ACPI_MUTEX_TYPE != ACPI_BINARY_SEMAPHORE)

200 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCreateMutex
201 ACPI_STATUS
202 AcpiOsCreateMutex (
203     ACPI_MUTEX              *OutHandle);
204 #endif

206 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsDeleteMutex
207 void
208 AcpiOsDeleteMutex (
209     ACPI_MUTEX              Handle);
210 #endif

212 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAcquireMutex
213 ACPI_STATUS
214 AcpiOsAcquireMutex (
215     ACPI_MUTEX              Handle,
216     UINT16                  Timeout);
217 #endif

219 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReleaseMutex
220 void
221 AcpiOsReleaseMutex (
222     ACPI_MUTEX              Handle);
223 #endif

225 #endif


228 /*
229  * Memory allocation and mapping
230  */
231 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAllocate
232 void *
233 AcpiOsAllocate (
234     ACPI_SIZE               Size);
235 #endif

237 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAllocateZeroed
238 void *
239 AcpiOsAllocateZeroed (
240     ACPI_SIZE               Size);
241 #endif

243 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsFree
244 void
245 AcpiOsFree (
246     void *                  Memory);
247 #endif

249 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsMapMemory
250 void *
251 AcpiOsMapMemory (
252     ACPI_PHYSICAL_ADDRESS   Where,
253     ACPI_SIZE               Length);
254 #endif

256 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsUnmapMemory
257 void
```

```
258 AcpiOsUnmapMemory (
259     void                    *LogicalAddress,
260     ACPI_SIZE               Size);
261 #endif

263 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetPhysicalAddress
264 ACPI_STATUS
265 AcpiOsGetPhysicalAddress (
266     void                    *LogicalAddress,
267     ACPI_PHYSICAL_ADDRESS   *PhysicalAddress);
268 #endif


271 /*
272  * Memory/Object Cache
273  */
274 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCreateCache
275 ACPI_STATUS
276 AcpiOsCreateCache (
277     char                    *CacheName,
278     UINT16                  ObjectSize,
279     UINT16                  MaxDepth,
280     ACPI_CACHE_T            **ReturnCache);
281 #endif

283 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsDeleteCache
284 ACPI_STATUS
285 AcpiOsDeleteCache (
286     ACPI_CACHE_T            *Cache);
287 #endif

289 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPurgeCache
290 ACPI_STATUS
291 AcpiOsPurgeCache (
292     ACPI_CACHE_T            *Cache);
293 #endif

295 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAcquireObject
296 void *
297 AcpiOsAcquireObject (
298     ACPI_CACHE_T            *Cache);
299 #endif

301 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReleaseObject
302 ACPI_STATUS
303 AcpiOsReleaseObject (
304     ACPI_CACHE_T            *Cache,
305     void                    *Object);
306 #endif


309 /*
310  * Interrupt handlers
311  */
312 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsInstallInterruptHandler
313 ACPI_STATUS
314 AcpiOsInstallInterruptHandler (
315     UINT32                  InterruptNumber,
316     ACPI_OSD_HANDLER        ServiceRoutine,
317     void                    *Context);
318 #endif

320 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsRemoveInterruptHandler
321 ACPI_STATUS
322 AcpiOsRemoveInterruptHandler (
323     UINT32                  InterruptNumber,
```



```
324     ACPI_OSD_HANDLER        ServiceRoutine);
325 #endif


328 /*
329  * Threads and Scheduling
330  */
331 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetThreadId
332 ACPI_THREAD_ID
333 AcpiOsGetThreadId (
334     void);
335 #endif

337 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsExecute
338 ACPI_STATUS
339 AcpiOsExecute (
340     ACPI_EXECUTE_TYPE       Type,
341     ACPI_OSD_EXEC_CALLBACK  Function,
342     void                    *Context);
343 #endif

345 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWaitEventsComplete
346 void
347 AcpiOsWaitEventsComplete (
348     void);
349 #endif

351 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsSleep
352 void
353 AcpiOsSleep (
354     UINT64                  Milliseconds);
355 #endif

357 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsStall
358 void
359 AcpiOsStall (
360     UINT32                  Microseconds);
361 #endif


364 /*
365  * Platform and hardware-independent I/O interfaces
366  */
367 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReadPort
368 ACPI_STATUS
369 AcpiOsReadPort (
370     ACPI_IO_ADDRESS         Address,
371     UINT32                  *Value,
372     UINT32                  Width);
373 #endif

375 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWritePort
376 ACPI_STATUS
377 AcpiOsWritePort (
378     ACPI_IO_ADDRESS         Address,
379     UINT32                  Value,
380     UINT32                  Width);
381 #endif


384 /*
385  * Platform and hardware-independent physical memory interfaces
386  */
387 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReadMemory
388 ACPI_STATUS
389 AcpiOsReadMemory (
```

```
390     ACPI_PHYSICAL_ADDRESS   Address,
391     UINT64                  *Value,
392     UINT32                  Width);
393 #endif

395 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWriteMemory
396 ACPI_STATUS
397 AcpiOsWriteMemory (
398     ACPI_PHYSICAL_ADDRESS   Address,
399     UINT64                  Value,
400     UINT32                  Width);
401 #endif


404 /*
405  * Platform and hardware-independent PCI configuration space access
406  * Note: Can't use "Register" as a parameter, changed to "Reg" --
407  * certain compilers complain.
408  */
409 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReadPciConfiguration
410 ACPI_STATUS
411 AcpiOsReadPciConfiguration (
412     ACPI_PCI_ID             *PciId,
413     UINT32                  Reg,
414     UINT64                  *Value,
415     UINT32                  Width);
416 #endif

418 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWritePciConfiguration
419 ACPI_STATUS
420 AcpiOsWritePciConfiguration (
421     ACPI_PCI_ID             *PciId,
422     UINT32                  Reg,
423     UINT64                  Value,
424     UINT32                  Width);
425 #endif


428 /*
429  * Miscellaneous
430  */
431 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReadable
432 BOOLEAN
433 AcpiOsReadable (
434     void                    *Pointer,
435     ACPI_SIZE               Length);
436 #endif

438 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWritable
439 BOOLEAN
440 AcpiOsWritable (
441     void                    *Pointer,
442     ACPI_SIZE               Length);
443 #endif

445 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTimer
446 UINT64
447 AcpiOsGetTimer (
448     void);
449 #endif

451 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsSignal
452 ACPI_STATUS
453 AcpiOsSignal (
454     UINT32                  Function,
455     void                    *Info);
```



```
456 #endif


459 /*
460  * Debug print routines
461  */
462 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPrintf
463 void ACPI_INTERNAL_VAR_XFACE
464 AcpiOsPrintf (
465     const char              *Format,
466     ...);
467 #endif

469 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsVprintf
470 void
471 AcpiOsVprintf (
472     const char              *Format,
473     va_list                 Args);
474 #endif

476 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsRedirectOutput
477 void
478 AcpiOsRedirectOutput (
479     void                    *Destination);
480 #endif


483 /*
484  * Debug input
485  */
486 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetLine
487 ACPI_STATUS
488 AcpiOsGetLine (
489     char                    *Buffer,
490     UINT32                  BufferLength,
491     UINT32                  *BytesRead);
492 #endif


495 /*
496  * Obtain ACPI table(s)
497  */
498 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByName
499 ACPI_STATUS
500 AcpiOsGetTableByName (
501     char                    *Signature,
502     UINT32                  Instance,
503     ACPI_TABLE_HEADER       **Table,
504     ACPI_PHYSICAL_ADDRESS   *Address);
505 #endif

507 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByIndex
508 ACPI_STATUS
509 AcpiOsGetTableByIndex (
510     UINT32                  Index,
511     ACPI_TABLE_HEADER       **Table,
512     UINT32                  *Instance,
513     ACPI_PHYSICAL_ADDRESS   *Address);
514 #endif

516 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByAddress
517 ACPI_STATUS
518 AcpiOsGetTableByAddress (
519     ACPI_PHYSICAL_ADDRESS   Address,
520     ACPI_TABLE_HEADER       **Table);
521 #endif
```

```
524 /*
525  * Directory manipulation
526  */
527 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsOpenDirectory
528 void *
529 AcpiOsOpenDirectory (
530     char                    *Pathname,
531     char                    *WildcardSpec,
532     char                    RequestedFileType);
533 #endif

535 /* RequesteFileType values */

537 #define REQUEST_FILE_ONLY                   0
538 #define REQUEST_DIR_ONLY                    1


541 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetNextFilename
542 char *
543 AcpiOsGetNextFilename (
544     void                    *DirHandle);
545 #endif

547 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCloseDirectory
548 void
549 AcpiOsCloseDirectory (
550     void                    *DirHandle);
551 #endif


554 #endif /* __ACPIOSXF_H__ */
```

```
**********************************************************
   20189 Thu Dec 26 13:49:53 2013
new/usr/src/common/acpica/include/acpixf.h
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acpixf.h - External interfaces to the ACPI subsystem
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#ifndef __ACXFACE_H__
#define __ACXFACE_H__

/* Current ACPICA subsystem version in YYYYMMDD format */

#define ACPI_CA_VERSION                 0x20131218

#include "acconfig.h"
#include "actypes.h"
#include "actbl.h"
#include "acbuffer.h"

/*
```



```
 * Globals that are publically available
 */
extern UINT32               AcpiCurrentGpeCount;
extern ACPI_TABLE_FADT      AcpiGbl_FADT;
extern BOOLEAN              AcpiGbl_SystemAwakeAndRunning;
extern BOOLEAN              AcpiGbl_ReducedHardware;        /* ACPI 5.0 */
extern UINT8                AcpiGbl_OsiData;

/* Runtime configuration of debug print levels */

extern UINT32               AcpiDbgLevel;
extern UINT32               AcpiDbgLayer;

/* ACPICA runtime options */

extern UINT8                AcpiGbl_AllMethodsSerialized;
extern UINT8                AcpiGbl_CopyDsdtLocally;
extern UINT8                AcpiGbl_CreateOsiMethod;
extern UINT8                AcpiGbl_DisableAutoRepair;
extern UINT8                AcpiGbl_DisableSsdtTableLoad;
extern UINT8                AcpiGbl_DoNotUseXsdt;
extern UINT8                AcpiGbl_EnableAmlDebugObject;
extern UINT8                AcpiGbl_EnableInterpreterSlack;
extern UINT32               AcpiGbl_TraceFlags;
extern ACPI_NAME            AcpiGbl_TraceMethodName;
extern UINT8                AcpiGbl_TruncateIoAddresses;
extern UINT8                AcpiGbl_Use32BitFadtAddresses;
extern UINT8                AcpiGbl_UseDefaultRegisterWidths;


/*
 * Hardware-reduced prototypes. All interfaces that use these macros will
 * be configured out of the ACPICA build if the ACPI_REDUCED_HARDWARE flag
 * is set to TRUE.
 */
#if (!ACPI_REDUCED_HARDWARE)
#define ACPI_HW_DEPENDENT_RETURN_STATUS(Prototype) \
    Prototype;

#define ACPI_HW_DEPENDENT_RETURN_OK(Prototype) \
    Prototype;

#define ACPI_HW_DEPENDENT_RETURN_VOID(Prototype) \
    Prototype;

#else
#define ACPI_HW_DEPENDENT_RETURN_STATUS(Prototype) \
    static ACPI_INLINE Prototype {return(AE_NOT_CONFIGURED);}

#define ACPI_HW_DEPENDENT_RETURN_OK(Prototype) \
    static ACPI_INLINE Prototype {return(AE_OK);}

#define ACPI_HW_DEPENDENT_RETURN_VOID(Prototype) \
    static ACPI_INLINE Prototype {return;}

#endif /* !ACPI_REDUCED_HARDWARE */


/*
 * Initialization
 */
ACPI_STATUS
AcpiInitializeTables (
    ACPI_TABLE_DESC         *InitialStorage,
    UINT32                  InitialTableCount,
    BOOLEAN                 AllowResize);
```

```
125 ACPI_STATUS
126 AcpiInitializeSubsystem (
127     void);

129 ACPI_STATUS
130 AcpiEnableSubsystem (
131     UINT32                  Flags);

133 ACPI_STATUS
134 AcpiInitializeObjects (
135     UINT32                  Flags);

137 ACPI_STATUS
138 AcpiTerminate (
139     void);


142 /*
143  * Miscellaneous global interfaces
144  */
145 ACPI_HW_DEPENDENT_RETURN_STATUS (
146 ACPI_STATUS
147 AcpiEnable (
148     void))

150 ACPI_HW_DEPENDENT_RETURN_STATUS (
151 ACPI_STATUS
152 AcpiDisable (
153     void))

155 ACPI_STATUS
156 AcpiSubsystemStatus (
157     void);

159 ACPI_STATUS
160 AcpiGetSystemInfo (
161     ACPI_BUFFER             *RetBuffer);

163 ACPI_STATUS
164 AcpiGetStatistics (
165     ACPI_STATISTICS         *Stats);

167 const char *
168 AcpiFormatException (
169     ACPI_STATUS             Exception);

171 ACPI_STATUS
172 AcpiPurgeCachedObjects (
173     void);

175 ACPI_STATUS
176 AcpiInstallInterface (
177     ACPI_STRING             InterfaceName);

179 ACPI_STATUS
180 AcpiRemoveInterface (
181     ACPI_STRING             InterfaceName);

183 ACPI_STATUS
184 AcpiUpdateInterfaces (
185     UINT8                   Action);

187 UINT32
188 AcpiCheckAddressRange (
189     ACPI_ADR_SPACE_TYPE     SpaceId,
```



```
190     ACPI_PHYSICAL_ADDRESS   Address,
191     ACPI_SIZE               Length,
192     BOOLEAN                 Warn);

194 ACPI_STATUS
195 AcpiDecodePldBuffer (
196     UINT8                   *InBuffer,
197     ACPI_SIZE               Length,
198     ACPI_PLD_INFO           **ReturnBuffer);


201 /*
202  * ACPI table load/unload interfaces
203  */
204 ACPI_STATUS
205 AcpiLoadTable (
206     ACPI_TABLE_HEADER       *Table);

208 ACPI_STATUS
209 AcpiUnloadParentTable (
210     ACPI_HANDLE             Object);

212 ACPI_STATUS
213 AcpiLoadTables (
214     void);


217 /*
218  * ACPI table manipulation interfaces
219  */
220 ACPI_STATUS
221 AcpiReallocateRootTable (
222     void);

224 ACPI_STATUS
225 AcpiFindRootPointer (
226     ACPI_SIZE               *RsdpAddress);

228 ACPI_STATUS
229 AcpiGetTableHeader (
230     ACPI_STRING             Signature,
231     UINT32                  Instance,
232     ACPI_TABLE_HEADER       *OutTableHeader);

234 ACPI_STATUS
235 AcpiGetTable (
236     ACPI_STRING             Signature,
237     UINT32                  Instance,
238     ACPI_TABLE_HEADER       **OutTable);

240 ACPI_STATUS
241 AcpiGetTableByIndex (
242     UINT32                  TableIndex,
243     ACPI_TABLE_HEADER       **OutTable);

245 ACPI_STATUS
246 AcpiInstallTableHandler (
247     ACPI_TABLE_HANDLER      Handler,
248     void                    *Context);

250 ACPI_STATUS
251 AcpiRemoveTableHandler (
252     ACPI_TABLE_HANDLER      Handler);


255 /*
```

```
 * Namespace and name interfaces
 */
ACPI_STATUS
AcpiWalkNamespace (
    ACPI_OBJECT_TYPE        Type,
    ACPI_HANDLE             StartObject,
    UINT32                  MaxDepth,
    ACPI_WALK_CALLBACK      DescendingCallback,
    ACPI_WALK_CALLBACK      AscendingCallback,
    void                    *Context,
    void                    **ReturnValue);

ACPI_STATUS
AcpiGetDevices (
    char                    *HID,
    ACPI_WALK_CALLBACK      UserFunction,
    void                    *Context,
    void                    **ReturnValue);

ACPI_STATUS
AcpiGetName (
    ACPI_HANDLE             Object,
    UINT32                  NameType,
    ACPI_BUFFER             *RetPathPtr);

ACPI_STATUS
AcpiGetHandle (
    ACPI_HANDLE             Parent,
    ACPI_STRING             Pathname,
    ACPI_HANDLE             *RetHandle);

ACPI_STATUS
AcpiAttachData (
    ACPI_HANDLE             Object,
    ACPI_OBJECT_HANDLER     Handler,
    void                    *Data);

ACPI_STATUS
AcpiDetachData (
    ACPI_HANDLE             Object,
    ACPI_OBJECT_HANDLER     Handler);

ACPI_STATUS
AcpiGetData (
    ACPI_HANDLE             Object,
    ACPI_OBJECT_HANDLER     Handler,
    void                    **Data);

ACPI_STATUS
AcpiDebugTrace (
    char                    *Name,
    UINT32                  DebugLevel,
    UINT32                  DebugLayer,
    UINT32                  Flags);


/*
 * Object manipulation and enumeration
 */
ACPI_STATUS
AcpiEvaluateObject (
    ACPI_HANDLE             Object,
    ACPI_STRING             Pathname,
    ACPI_OBJECT_LIST        *ParameterObjects,
    ACPI_BUFFER             *ReturnObjectBuffer);
```



```
ACPI_STATUS
AcpiEvaluateObjectTyped (
    ACPI_HANDLE             Object,
    ACPI_STRING             Pathname,
    ACPI_OBJECT_LIST        *ExternalParams,
    ACPI_BUFFER             *ReturnBuffer,
    ACPI_OBJECT_TYPE        ReturnType);

ACPI_STATUS
AcpiGetObjectInfo (
    ACPI_HANDLE             Object,
    ACPI_DEVICE_INFO        **ReturnBuffer);

ACPI_STATUS
AcpiInstallMethod (
    UINT8                   *Buffer);

ACPI_STATUS
AcpiGetNextObject (
    ACPI_OBJECT_TYPE        Type,
    ACPI_HANDLE             Parent,
    ACPI_HANDLE             Child,
    ACPI_HANDLE             *OutHandle);

ACPI_STATUS
AcpiGetType (
    ACPI_HANDLE             Object,
    ACPI_OBJECT_TYPE        *OutType);

ACPI_STATUS
AcpiGetParent (
    ACPI_HANDLE             Object,
    ACPI_HANDLE             *OutHandle);


/*
 * Handler interfaces
 */
ACPI_STATUS
AcpiInstallInitializationHandler (
    ACPI_INIT_HANDLER       Handler,
    UINT32                  Function);

ACPI_HW_DEPENDENT_RETURN_STATUS (
ACPI_STATUS
AcpiInstallSciHandler (
    ACPI_SCI_HANDLER        Address,
    void                    *Context))

ACPI_HW_DEPENDENT_RETURN_STATUS (
ACPI_STATUS
AcpiRemoveSciHandler (
    ACPI_SCI_HANDLER        Address))

ACPI_HW_DEPENDENT_RETURN_STATUS (
ACPI_STATUS
AcpiInstallGlobalEventHandler (
    ACPI_GBL_EVENT_HANDLER  Handler,
    void                    *Context))

ACPI_HW_DEPENDENT_RETURN_STATUS (
ACPI_STATUS
AcpiInstallFixedEventHandler (
    UINT32                  AcpiEvent,
    ACPI_EVENT_HANDLER      Handler,
    void                    *Context))
```

```
389 ACPI_HW_DEPENDENT_RETURN_STATUS (
390 ACPI_STATUS
391 AcpiRemoveFixedEventHandler (
392     UINT32                  AcpiEvent,
393     ACPI_EVENT_HANDLER      Handler))

395 ACPI_HW_DEPENDENT_RETURN_STATUS (
396 ACPI_STATUS
397 AcpiInstallGpeHandler (
398     ACPI_HANDLE             GpeDevice,
399     UINT32                  GpeNumber,
400     UINT32                  Type,
401     ACPI_GPE_HANDLER        Address,
402     void                    *Context))

404 ACPI_HW_DEPENDENT_RETURN_STATUS (
405 ACPI_STATUS
406 AcpiRemoveGpeHandler (
407     ACPI_HANDLE             GpeDevice,
408     UINT32                  GpeNumber,
409     ACPI_GPE_HANDLER        Address))

411 ACPI_STATUS
412 AcpiInstallNotifyHandler (
413     ACPI_HANDLE             Device,
414     UINT32                  HandlerType,
415     ACPI_NOTIFY_HANDLER     Handler,
416     void                    *Context);

418 ACPI_STATUS
419 AcpiRemoveNotifyHandler (
420     ACPI_HANDLE             Device,
421     UINT32                  HandlerType,
422     ACPI_NOTIFY_HANDLER     Handler);

424 ACPI_STATUS
425 AcpiInstallAddressSpaceHandler (
426     ACPI_HANDLE             Device,
427     ACPI_ADR_SPACE_TYPE     SpaceId,
428     ACPI_ADR_SPACE_HANDLER  Handler,
429     ACPI_ADR_SPACE_SETUP    Setup,
430     void                    *Context);

432 ACPI_STATUS
433 AcpiRemoveAddressSpaceHandler (
434     ACPI_HANDLE             Device,
435     ACPI_ADR_SPACE_TYPE     SpaceId,
436     ACPI_ADR_SPACE_HANDLER  Handler);

438 ACPI_STATUS
439 AcpiInstallExceptionHandler (
440     ACPI_EXCEPTION_HANDLER  Handler);

442 ACPI_STATUS
443 AcpiInstallInterfaceHandler (
444     ACPI_INTERFACE_HANDLER  Handler);


447 /*
448  * Global Lock interfaces
449  */
450 ACPI_HW_DEPENDENT_RETURN_STATUS (
451 ACPI_STATUS
452 AcpiAcquireGlobalLock (
453     UINT16                  Timeout,
```



```
454     UINT32                  *Handle))

456 ACPI_HW_DEPENDENT_RETURN_STATUS (
457 ACPI_STATUS
458 AcpiReleaseGlobalLock (
459     UINT32                  Handle))


462 /*
463  * Interfaces to AML mutex objects
464  */
465 ACPI_STATUS
466 AcpiAcquireMutex (
467     ACPI_HANDLE             Handle,
468     ACPI_STRING             Pathname,
469     UINT16                  Timeout);

471 ACPI_STATUS
472 AcpiReleaseMutex (
473     ACPI_HANDLE             Handle,
474     ACPI_STRING             Pathname);


477 /*
478  * Fixed Event interfaces
479  */
480 ACPI_HW_DEPENDENT_RETURN_STATUS (
481 ACPI_STATUS
482 AcpiEnableEvent (
483     UINT32                  Event,
484     UINT32                  Flags))

486 ACPI_HW_DEPENDENT_RETURN_STATUS (
487 ACPI_STATUS
488 AcpiDisableEvent (
489     UINT32                  Event,
490     UINT32                  Flags))

492 ACPI_HW_DEPENDENT_RETURN_STATUS (
493 ACPI_STATUS
494 AcpiClearEvent (
495     UINT32                  Event))

497 ACPI_HW_DEPENDENT_RETURN_STATUS (
498 ACPI_STATUS
499 AcpiGetEventStatus (
500     UINT32                  Event,
501     ACPI_EVENT_STATUS       *EventStatus))


504 /*
505  * General Purpose Event (GPE) Interfaces
506  */
507 ACPI_HW_DEPENDENT_RETURN_STATUS (
508 ACPI_STATUS
509 AcpiUpdateAllGpes (
510     void))

512 ACPI_HW_DEPENDENT_RETURN_STATUS (
513 ACPI_STATUS
514 AcpiEnableGpe (
515     ACPI_HANDLE             GpeDevice,
516     UINT32                  GpeNumber))

518 ACPI_HW_DEPENDENT_RETURN_STATUS (
519 ACPI_STATUS
```

```
520 AcpiDisableGpe (
521     ACPI_HANDLE             GpeDevice,
522     UINT32                  GpeNumber))

524 ACPI_HW_DEPENDENT_RETURN_STATUS (
525 ACPI_STATUS
526 AcpiClearGpe (
527     ACPI_HANDLE             GpeDevice,
528     UINT32                  GpeNumber))

530 ACPI_HW_DEPENDENT_RETURN_STATUS (
531 ACPI_STATUS
532 AcpiSetGpe (
533     ACPI_HANDLE             GpeDevice,
534     UINT32                  GpeNumber,
535     UINT8                   Action))

537 ACPI_HW_DEPENDENT_RETURN_STATUS (
538 ACPI_STATUS
539 AcpiFinishGpe (
540     ACPI_HANDLE             GpeDevice,
541     UINT32                  GpeNumber))

543 ACPI_HW_DEPENDENT_RETURN_STATUS (
544 ACPI_STATUS
545 AcpiSetupGpeForWake (
546     ACPI_HANDLE             ParentDevice,
547     ACPI_HANDLE             GpeDevice,
548     UINT32                  GpeNumber))

550 ACPI_HW_DEPENDENT_RETURN_STATUS (
551 ACPI_STATUS
552 AcpiSetGpeWakeMask (
553     ACPI_HANDLE             GpeDevice,
554     UINT32                  GpeNumber,
555     UINT8                   Action))

557 ACPI_HW_DEPENDENT_RETURN_STATUS (
558 ACPI_STATUS
559 AcpiGetGpeStatus (
560     ACPI_HANDLE             GpeDevice,
561     UINT32                  GpeNumber,
562     ACPI_EVENT_STATUS       *EventStatus))

564 ACPI_HW_DEPENDENT_RETURN_STATUS (
565 ACPI_STATUS
566 AcpiDisableAllGpes (
567     void))

569 ACPI_HW_DEPENDENT_RETURN_STATUS (
570 ACPI_STATUS
571 AcpiEnableAllRuntimeGpes (
572     void))

574 ACPI_HW_DEPENDENT_RETURN_STATUS (
575 ACPI_STATUS
576 AcpiGetGpeDevice (
577     UINT32                  GpeIndex,
578     ACPI_HANDLE             *GpeDevice))

580 ACPI_HW_DEPENDENT_RETURN_STATUS (
581 ACPI_STATUS
582 AcpiInstallGpeBlock (
583     ACPI_HANDLE             GpeDevice,
584     ACPI_GENERIC_ADDRESS    *GpeBlockAddress,
585     UINT32                  RegisterCount,
```



```
586     UINT32                  InterruptNumber))

588 ACPI_HW_DEPENDENT_RETURN_STATUS (
589 ACPI_STATUS
590 AcpiRemoveGpeBlock (
591     ACPI_HANDLE             GpeDevice))


594 /*
595  * Resource interfaces
596  */
597 typedef
598 ACPI_STATUS (*ACPI_WALK_RESOURCE_CALLBACK) (
599     ACPI_RESOURCE           *Resource,
600     void                    *Context);

602 ACPI_STATUS
603 AcpiGetVendorResource (
604     ACPI_HANDLE             Device,
605     char                    *Name,
606     ACPI_VENDOR_UUID        *Uuid,
607     ACPI_BUFFER             *RetBuffer);

609 ACPI_STATUS
610 AcpiGetCurrentResources (
611     ACPI_HANDLE             Device,
612     ACPI_BUFFER             *RetBuffer);

614 ACPI_STATUS
615 AcpiGetPossibleResources (
616     ACPI_HANDLE             Device,
617     ACPI_BUFFER             *RetBuffer);

619 ACPI_STATUS
620 AcpiGetEventResources (
621     ACPI_HANDLE             DeviceHandle,
622     ACPI_BUFFER             *RetBuffer);

624 ACPI_STATUS
625 AcpiWalkResourceBuffer (
626     ACPI_BUFFER                 *Buffer,
627     ACPI_WALK_RESOURCE_CALLBACK UserFunction,
628     void                        *Context);

630 ACPI_STATUS
631 AcpiWalkResources (
632     ACPI_HANDLE                 Device,
633     char                        *Name,
634     ACPI_WALK_RESOURCE_CALLBACK UserFunction,
635     void                        *Context);

637 ACPI_STATUS
638 AcpiSetCurrentResources (
639     ACPI_HANDLE             Device,
640     ACPI_BUFFER             *InBuffer);

642 ACPI_STATUS
643 AcpiGetIrqRoutingTable (
644     ACPI_HANDLE             Device,
645     ACPI_BUFFER             *RetBuffer);

647 ACPI_STATUS
648 AcpiResourceToAddress64 (
649     ACPI_RESOURCE           *Resource,
650     ACPI_RESOURCE_ADDRESS64 *Out);
```

```
652 ACPI_STATUS
653 AcpiBufferToResource (
654     UINT8                   *AmlBuffer,
655     UINT16                  AmlBufferLength,
656     ACPI_RESOURCE           **ResourcePtr);


659 /*
660  * Hardware (ACPI device) interfaces
661  */
662 ACPI_STATUS
663 AcpiReset (
664     void);

666 ACPI_STATUS
667 AcpiRead (
668     UINT64                  *Value,
669     ACPI_GENERIC_ADDRESS    *Reg);

671 ACPI_STATUS
672 AcpiWrite (
673     UINT64                  Value,
674     ACPI_GENERIC_ADDRESS    *Reg);

676 ACPI_HW_DEPENDENT_RETURN_STATUS (
677 ACPI_STATUS
678 AcpiReadBitRegister (
679     UINT32                  RegisterId,
680     UINT32                  *ReturnValue))

682 ACPI_HW_DEPENDENT_RETURN_STATUS (
683 ACPI_STATUS
684 AcpiWriteBitRegister (
685     UINT32                  RegisterId,
686     UINT32                  Value))


689 /*
690  * Sleep/Wake interfaces
691  */
692 ACPI_STATUS
693 AcpiGetSleepTypeData (
694     UINT8                   SleepState,
695     UINT8                   *Slp_TypA,
696     UINT8                   *Slp_TypB);

698 ACPI_STATUS
699 AcpiEnterSleepStatePrep (
700     UINT8                   SleepState);

702 ACPI_STATUS
703 AcpiEnterSleepState (
704     UINT8                   SleepState);

706 ACPI_HW_DEPENDENT_RETURN_STATUS (
707 ACPI_STATUS
708 AcpiEnterSleepStateS4bios (
709     void))

711 ACPI_STATUS
712 AcpiLeaveSleepStatePrep (
713     UINT8                   SleepState);

715 ACPI_STATUS
716 AcpiLeaveSleepState (
717     UINT8                   SleepState);
```



```
719 ACPI_HW_DEPENDENT_RETURN_STATUS (
720 ACPI_STATUS
721 AcpiSetFirmwareWakingVector (
722     UINT32                  PhysicalAddress))

724 #if ACPI_MACHINE_WIDTH == 64
725 ACPI_HW_DEPENDENT_RETURN_STATUS (
726 ACPI_STATUS
727 AcpiSetFirmwareWakingVector64 (
728     UINT64                  PhysicalAddress))
729 #endif


732 /*
733  * ACPI Timer interfaces
734  */
735 ACPI_HW_DEPENDENT_RETURN_STATUS (
736 ACPI_STATUS
737 AcpiGetTimerResolution (
738     UINT32                  *Resolution))

740 ACPI_HW_DEPENDENT_RETURN_STATUS (
741 ACPI_STATUS
742 AcpiGetTimer (
743     UINT32                  *Ticks))

745 ACPI_HW_DEPENDENT_RETURN_STATUS (
746 ACPI_STATUS
747 AcpiGetTimerDuration (
748     UINT32                  StartTicks,
749     UINT32                  EndTicks,
750     UINT32                  *TimeElapsed))


753 /*
754  * Error/Warning output
755  */
756 ACPI_PRINTF_LIKE(3)
757 void ACPI_INTERNAL_VAR_XFACE
758 AcpiError (
759     const char              *ModuleName,
760     UINT32                  LineNumber,
761     const char              *Format,
762     ...);

764 ACPI_PRINTF_LIKE(4)
765 void  ACPI_INTERNAL_VAR_XFACE
766 AcpiException (
767     const char              *ModuleName,
768     UINT32                  LineNumber,
769     ACPI_STATUS             Status,
770     const char              *Format,
771     ...);

773 ACPI_PRINTF_LIKE(3)
774 void ACPI_INTERNAL_VAR_XFACE
775 AcpiWarning (
776     const char              *ModuleName,
777     UINT32                  LineNumber,
778     const char              *Format,
779     ...);

781 ACPI_PRINTF_LIKE(3)
782 void ACPI_INTERNAL_VAR_XFACE
783 AcpiInfo (
```

```
784     const char              *ModuleName,
785     UINT32                  LineNumber,
786     const char              *Format,
787     ...);

789 ACPI_PRINTF_LIKE(3)
790 void ACPI_INTERNAL_VAR_XFACE
791 AcpiBiosError (
792     const char              *ModuleName,
793     UINT32                  LineNumber,
794     const char              *Format,
795     ...);

797 ACPI_PRINTF_LIKE(3)
798 void ACPI_INTERNAL_VAR_XFACE
799 AcpiBiosWarning (
800     const char              *ModuleName,
801     UINT32                  LineNumber,
802     const char              *Format,
803     ...);


806 /*
807  * Debug output
808  */
809 #ifdef ACPI_DEBUG_OUTPUT

811 ACPI_PRINTF_LIKE(6)
812 void ACPI_INTERNAL_VAR_XFACE
813 AcpiDebugPrint (
814     UINT32                  RequestedDebugLevel,
815     UINT32                  LineNumber,
816     const char              *FunctionName,
817     const char              *ModuleName,
818     UINT32                  ComponentId,
819     const char              *Format,
820     ...);

822 ACPI_PRINTF_LIKE(6)
823 void ACPI_INTERNAL_VAR_XFACE
824 AcpiDebugPrintRaw (
825     UINT32                  RequestedDebugLevel,
826     UINT32                  LineNumber,
827     const char              *FunctionName,
828     const char              *ModuleName,
829     UINT32                  ComponentId,
830     const char              *Format,
831     ...);
832 #endif

834 #endif /* __ACXFACE_H__ */
```

```
**********************************************************
   42485 Thu Dec 26 13:49:54 2013
new/usr/src/common/acpica/include/acpredef.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acpredef - Information table for ACPI predefined methods and objects
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACPREDEF_H__
#define __ACPREDEF_H__


/******************************************************************************
 *
 * Return Package types
 *
 * 1) PTYPE1 packages do not contain sub-packages.
 *
 * ACPI_PTYPE1_FIXED: Fixed-length length, 1 or 2 object types:
 *      object type
 *      count
 *      object type
 *      count
 *
 * ACPI_PTYPE1_VAR: Variable-length length. Zero-length package is allowed:
```



```
 *      object type (Int/Buf/Ref)
 *
 * ACPI_PTYPE1_OPTION: Package has some required and some optional elements
 *      (Used for _PRW)
 *
 *
 * 2) PTYPE2 packages contain a Variable-length number of sub-packages. Each
 *    of the different types describe the contents of each of the sub-packages.
 *
 * ACPI_PTYPE2: Each subpackage contains 1 or 2 object types. Zero-length
 *      parent package is allowed:
 *      object type
 *      count
 *      object type
 *      count
 *      (Used for _ALR,_MLS,_PSS,_TRT,_TSS)
 *
 * ACPI_PTYPE2_COUNT: Each subpackage has a count as first element.
 *      Zero-length parent package is allowed:
 *      object type
 *      (Used for _CSD,_PSD,_TSD)
 *
 * ACPI_PTYPE2_PKG_COUNT: Count of subpackages at start, 1 or 2 object types:
 *      object type
 *      count
 *      object type
 *      count
 *      (Used for _CST)
 *
 * ACPI_PTYPE2_FIXED: Each subpackage is of Fixed-length. Zero-length
 *      parent package is allowed.
 *      (Used for _PRT)
 *
 * ACPI_PTYPE2_MIN: Each subpackage has a Variable-length but minimum length.
 *      Zero-length parent package is allowed:
 *      (Used for _HPX)
 *
 * ACPI_PTYPE2_REV_FIXED: Revision at start, each subpackage is Fixed-length
 *      (Used for _ART, _FPS)
 *
 * ACPI_PTYPE2_FIX_VAR: Each subpackage consists of some fixed-length elements
 *      followed by an optional element. Zero-length parent package is allowed.
 *      object type
 *      count
 *      object type
 *      count = 0 (optional)
 *      (Used for _DLM)
 *
 *****************************************************************************/

enum AcpiReturnPackageTypes
{
    ACPI_PTYPE1_FIXED       = 1,
    ACPI_PTYPE1_VAR         = 2,
    ACPI_PTYPE1_OPTION      = 3,
    ACPI_PTYPE2             = 4,
    ACPI_PTYPE2_COUNT       = 5,
    ACPI_PTYPE2_PKG_COUNT   = 6,
    ACPI_PTYPE2_FIXED       = 7,
    ACPI_PTYPE2_MIN         = 8,
    ACPI_PTYPE2_REV_FIXED   = 9,
    ACPI_PTYPE2_FIX_VAR     = 10
};


/* Support macros for users of the predefined info table */
```

```
#define METHOD_PREDEF_ARGS_MAX          4
#define METHOD_ARG_BIT_WIDTH            3
#define METHOD_ARG_MASK                 0x0007
#define ARG_COUNT_IS_MINIMUM            0x8000
#define METHOD_MAX_ARG_TYPE             ACPI_TYPE_PACKAGE

#define METHOD_GET_ARG_COUNT(ArgList)   ((ArgList) & METHOD_ARG_MASK)
#define METHOD_GET_NEXT_TYPE(ArgList)   (((ArgList) >>= METHOD_ARG_BIT_WIDTH) &

/* Macros used to build the predefined info table */

#define METHOD_0ARGS                    0
#define METHOD_1ARGS(a1)                (1 | (a1 << 3))
#define METHOD_2ARGS(a1,a2)             (2 | (a1 << 3) | (a2 << 6))
#define METHOD_3ARGS(a1,a2,a3)          (3 | (a1 << 3) | (a2 << 6) | (a3 << 9))
#define METHOD_4ARGS(a1,a2,a3,a4)       (4 | (a1 << 3) | (a2 << 6) | (a3 << 9) |

#define METHOD_RETURNS(type)            (type)
#define METHOD_NO_RETURN_VALUE          0

#define PACKAGE_INFO(a,b,c,d,e,f)       {{{(a),(b),(c),(d)}, ((((UINT16)(f)) <<


/* Support macros for the resource descriptor info table */

#define WIDTH_1                         0x0001
#define WIDTH_2                         0x0002
#define WIDTH_3                         0x0004
#define WIDTH_8                         0x0008
#define WIDTH_16                        0x0010
#define WIDTH_32                        0x0020
#define WIDTH_64                        0x0040
#define VARIABLE_DATA                   0x0080
#define NUM_RESOURCE_WIDTHS             8

#define WIDTH_ADDRESS                   WIDTH_16 | WIDTH_32 | WIDTH_64


#ifdef ACPI_CREATE_PREDEFINED_TABLE
/******************************************************************************
 *
 * Predefined method/object information table.
 *
 * These are the names that can actually be evaluated via AcpiEvaluateObject.
 * Not present in this table are the following:
 *
 *      1) Predefined/Reserved names that are never evaluated via
 *         AcpiEvaluateObject:
 *              _Lxx and _Exx GPE methods
 *              _Qxx EC methods
 *              _T_x compiler temporary variables
 *              _Wxx wake events
 *
 *      2) Predefined names that never actually exist within the AML code:
 *              Predefined resource descriptor field names
 *
 *      3) Predefined names that are implemented within ACPICA:
 *              _OSI
 *
 * The main entries in the table each contain the following items:
 *
 * Name                 - The ACPI reserved name
 * ArgumentList         - Contains (in 16 bits), the number of required
 *                        arguments to the method (3 bits), and a 3-bit type
 *                        field for each argument (up to 4 arguments). The
```



```
 *                        METHOD_?ARGS macros generate the correct packed data.
 * ExpectedBtypes       - Allowed type(s) for the return value.
 *                        0 means that no return value is expected.
 *
 * For methods that return packages, the next entry in the table contains
 * information about the expected structure of the package. This information
 * is saved here (rather than in a separate table) in order to minimize the
 * overall size of the stored data.
 *
 * Note: The additional braces are intended to promote portability.
 *
 * Note2: Table is used by the kernel-resident subsystem, the iASL compiler,
 * and the AcpiHelp utility.
 *
 * TBD: _PRT - currently ignore reversed entries. Attempt to fix in nsrepair.
 * Possibly fixing package elements like _BIF, etc.
 *
 *****************************************************************************/

const ACPI_PREDEFINED_INFO          AcpiGbl_PredefinedMethods[] =
{
    {{"_AC0",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC1",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC2",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC3",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC4",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC5",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC6",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC7",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC8",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AC9",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_ADR",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_AEI",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

    {{"_AL0",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
                    PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

    {{"_AL1",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
                    PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

    {{"_AL2",   METHOD_0ARGS,
```

```
259                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
260                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

262     {{"_AL3",   METHOD_0ARGS,
263                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
264                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

266     {{"_AL4",   METHOD_0ARGS,
267                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
268                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

270     {{"_AL5",   METHOD_0ARGS,
271                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
272                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

274     {{"_AL6",   METHOD_0ARGS,
275                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
276                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

278     {{"_AL7",   METHOD_0ARGS,
279                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
280                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

282     {{"_AL8",   METHOD_0ARGS,
283                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
284                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

286     {{"_AL9",   METHOD_0ARGS,
287                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
288                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

290     {{"_ALC",   METHOD_0ARGS,
291                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

293     {{"_ALI",   METHOD_0ARGS,
294                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

296     {{"_ALP",   METHOD_0ARGS,
297                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

299     {{"_ALR",   METHOD_0ARGS,
300                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
301                     PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_INTEGER, 2,0,0,0),

303     {{"_ALT",   METHOD_0ARGS,
304                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

306     {{"_ART",   METHOD_0ARGS,
307                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (1 Int
308                     PACKAGE_INFO (ACPI_PTYPE2_REV_FIXED, ACPI_RTYPE_REFERENCE, 2

310     {{"_BBN",   METHOD_0ARGS,
311                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

313     {{"_BCL",   METHOD_0ARGS,
314                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Ints)
315                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 0,0,0,0),

317     {{"_BCM",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
318                 METHOD_NO_RETURN_VALUE}},

320     {{"_BCT",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
321                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

323     {{"_BDN",   METHOD_0ARGS,
324                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
```



```
326     {{"_BFS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
327                 METHOD_NO_RETURN_VALUE}},

329     {{"_BIF",   METHOD_0ARGS,
330                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (9 Int),(
331                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 9, ACPI

333     {{"_BIX",   METHOD_0ARGS,
334                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (16 Int),
335                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 16, ACP

337     {{"_BLT",   METHOD_3ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER, ACPI_TYPE_IN
338                 METHOD_NO_RETURN_VALUE}},

340     {{"_BMA",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
341                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

343     {{"_BMC",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
344                 METHOD_NO_RETURN_VALUE}},

346     {{"_BMD",   METHOD_0ARGS,
347                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (5 Int) *
348                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 5,0,0,0

350     {{"_BMS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
351                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

353     {{"_BQC",   METHOD_0ARGS,
354                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

356     {{"_BST",   METHOD_0ARGS,
357                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (4 Int) *
358                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 4,0,0,0

360     {{"_BTM",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
361                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

363     {{"_BTP",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
364                 METHOD_NO_RETURN_VALUE}},

366     {{"_CBA",   METHOD_0ARGS,
367                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* See PCI firmware spec

369     {{"_CDM",   METHOD_0ARGS,
370                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

372     {{"_CID",   METHOD_0ARGS,
373                 METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RT
374                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER | ACPI_RTY

376     {{"_CLS",   METHOD_0ARGS,
377                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (3 Int) *
378                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 3,0,0,0

380     {{"_CPC",   METHOD_0ARGS,
381                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Ints/
382                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER | ACPI_RTY

384     {{"_CRS",   METHOD_0ARGS,
385                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

387     {{"_CRT",   METHOD_0ARGS,
388                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

390     {{"_CSD",   METHOD_0ARGS,
```

```
391                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (1 Int
392                     PACKAGE_INFO (ACPI_PTYPE2_COUNT, ACPI_RTYPE_INTEGER, 0,0,0,0

394     {{"_CST",   METHOD_0ARGS,
395                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (1 Int
396                     PACKAGE_INFO (ACPI_PTYPE2_PKG_COUNT,ACPI_RTYPE_BUFFER, 1, AC

398     {{"_CWS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
399                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

401     {{"_DCK",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
402                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

404     {{"_DCS",   METHOD_0ARGS,
405                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

407     {{"_DDC",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
408                 METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_BUFFER)}},

410     {{"_DDN",   METHOD_0ARGS,
411                 METHOD_RETURNS (ACPI_RTYPE_STRING)}},

413     {{"_DEP",   METHOD_0ARGS,
414                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
415                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

417     {{"_DGS",   METHOD_0ARGS,
418                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

420     {{"_DIS",   METHOD_0ARGS,
421                 METHOD_NO_RETURN_VALUE}},

423     {{"_DLM",   METHOD_0ARGS,
424                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
425                     PACKAGE_INFO (ACPI_PTYPE2_FIX_VAR, ACPI_RTYPE_REFERENCE, 1,

427     {{"_DMA",   METHOD_0ARGS,
428                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

430     {{"_DOD",   METHOD_0ARGS,
431                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Ints)
432                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 0,0,0,0),

434     {{"_DOS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
435                 METHOD_NO_RETURN_VALUE}},

437     {{"_DSM",   METHOD_4ARGS (ACPI_TYPE_BUFFER, ACPI_TYPE_INTEGER, ACPI_TYPE_INT
438                 METHOD_RETURNS (ACPI_RTYPE_ALL)}}, /* Must return a value, but i

440     {{"_DSS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
441                 METHOD_NO_RETURN_VALUE}},

443     {{"_DSW",   METHOD_3ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER, ACPI_TYPE_IN
444                 METHOD_NO_RETURN_VALUE}},

446     {{"_DTI",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
447                 METHOD_NO_RETURN_VALUE}},

449     {{"_EC_",   METHOD_0ARGS,
450                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

452     {{"_EDL",   METHOD_0ARGS,
453                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
454                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

456     {{"_EJ0",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
```



```
457                 METHOD_NO_RETURN_VALUE}},

459     {{"_EJ1",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
460                 METHOD_NO_RETURN_VALUE}},

462     {{"_EJ2",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
463                 METHOD_NO_RETURN_VALUE}},

465     {{"_EJ3",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
466                 METHOD_NO_RETURN_VALUE}},

468     {{"_EJ4",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
469                 METHOD_NO_RETURN_VALUE}},

471     {{"_EJD",   METHOD_0ARGS,
472                 METHOD_RETURNS (ACPI_RTYPE_STRING)}},

474     {{"_ERR",   METHOD_3ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_STRING, ACPI_TYPE_INT
475                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* Internal use only, use

477     {{"_EVT",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
478                 METHOD_NO_RETURN_VALUE}},

480     {{"_FDE",   METHOD_0ARGS,
481                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

483     {{"_FDI",   METHOD_0ARGS,
484                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (16 Int)
485                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 16,0,0,

487     {{"_FDM",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
488                 METHOD_NO_RETURN_VALUE}},

490     {{"_FIF",   METHOD_0ARGS,
491                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (4 Int) *
492                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 4,0,0,0

494     {{"_FIX",   METHOD_0ARGS,
495                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Ints)
496                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 0,0,0,0),

498     {{"_FPS",   METHOD_0ARGS,
499                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (1 Int
500                     PACKAGE_INFO (ACPI_PTYPE2_REV_FIXED,ACPI_RTYPE_INTEGER, 5, 0

502     {{"_FSL",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
503                 METHOD_NO_RETURN_VALUE}},

505     {{"_FST",   METHOD_0ARGS,
506                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (3 Int) *
507                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 3,0,0,0

509     {{"_GAI",   METHOD_0ARGS,
510                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

512     {{"_GCP",   METHOD_0ARGS,
513                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

515     {{"_GHL",   METHOD_0ARGS,
516                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

518     {{"_GLK",   METHOD_0ARGS,
519                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

521     {{"_GPD",   METHOD_0ARGS,
522                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
```

```
524     {{"_GPE",   METHOD_0ARGS,
525                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* _GPE method, not _GPE
527     {{"_GRT",   METHOD_0ARGS,
528                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
530     {{"_GSB",   METHOD_0ARGS,
531                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
533     {{"_GTF",   METHOD_0ARGS,
534                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
536     {{"_GTM",   METHOD_0ARGS,
537                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
539     {{"_GTS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
540                 METHOD_NO_RETURN_VALUE}},
542     {{"_GWS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
543                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
545     {{"_HID",   METHOD_0ARGS,
546                 METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING)}},
548     {{"_HOT",   METHOD_0ARGS,
549                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
551     {{"_HPP",   METHOD_0ARGS,
552                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (4 Int) *
553                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 4,0,0,0
555     /*
556      * For _HPX, a single package is returned, containing a variable-length numb
557      * of sub-packages. Each sub-package contains a PCI record setting.
558      * There are several different type of record settings, of different
559      * lengths, but all elements of all settings are Integers.
560      */
561     {{"_HPX",   METHOD_0ARGS,
562                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
563                     PACKAGE_INFO (ACPI_PTYPE2_MIN, ACPI_RTYPE_INTEGER, 5,0,0,0),
565     {{"_HRV",   METHOD_0ARGS,
566                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
568     {{"_IFT",   METHOD_0ARGS,
569                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* See IPMI spec */
571     {{"_INI",   METHOD_0ARGS,
572                 METHOD_NO_RETURN_VALUE}},
574     {{"_IRC",   METHOD_0ARGS,
575                 METHOD_NO_RETURN_VALUE}},
577     {{"_LCK",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
578                 METHOD_NO_RETURN_VALUE}},
580     {{"_LID",   METHOD_0ARGS,
581                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
583     {{"_MAT",   METHOD_0ARGS,
584                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
586     {{"_MBM",   METHOD_0ARGS,
587                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (8 Int) *
588                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 8,0,0,0
```



```
590     {{"_MLS",   METHOD_0ARGS,
591                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
592                     PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_STRING, 1, ACPI_RTYPE_
594     {{"_MSG",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
595                 METHOD_NO_RETURN_VALUE}},
597     {{"_MSM",   METHOD_4ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER, ACPI_TYPE_IN
598                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
600     {{"_NTT",   METHOD_0ARGS,
601                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
603     {{"_OFF",   METHOD_0ARGS,
604                 METHOD_NO_RETURN_VALUE}},
606     {{"_ON_",   METHOD_0ARGS,
607                 METHOD_NO_RETURN_VALUE}},
609     {{"_OS_",   METHOD_0ARGS,
610                 METHOD_RETURNS (ACPI_RTYPE_STRING)}},
612     {{"_OSC",   METHOD_4ARGS (ACPI_TYPE_BUFFER, ACPI_TYPE_INTEGER, ACPI_TYPE_INT
613                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
615     {{"_OST",   METHOD_3ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER, ACPI_TYPE_BU
616                 METHOD_NO_RETURN_VALUE}},
618     {{"_PAI",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
619                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
621     {{"_PCL",   METHOD_0ARGS,
622                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
623                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0
625     {{"_PCT",   METHOD_0ARGS,
626                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (2 Buf) *
627                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_BUFFER, 2,0,0,0)
629     {{"_PDC",   METHOD_1ARGS (ACPI_TYPE_BUFFER),
630                 METHOD_NO_RETURN_VALUE}},
632     {{"_PDL",   METHOD_0ARGS,
633                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
635     {{"_PIC",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
636                 METHOD_NO_RETURN_VALUE}},
638     {{"_PIF",   METHOD_0ARGS,
639                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (3 Int),(
640                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 3, ACPI
642     {{"_PLD",   METHOD_0ARGS,
643                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Bufs)
644                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_BUFFER, 0,0,0,0),
646     {{"_PMC",   METHOD_0ARGS,
647                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (11 Int),
648                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 11, ACP
650     {{"_PMD",   METHOD_0ARGS,
651                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
652                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0
654     {{"_PMM",   METHOD_0ARGS,
```

```
655                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

657     {{"_PPC",   METHOD_0ARGS,
658                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

660     {{"_PPE",   METHOD_0ARGS,
661                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* See dig64 spec */

663     {{"_PR0",   METHOD_0ARGS,
664                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
665                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

667     {{"_PR1",   METHOD_0ARGS,
668                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
669                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

671     {{"_PR2",   METHOD_0ARGS,
672                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
673                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

675     {{"_PR3",   METHOD_0ARGS,
676                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
677                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

679     {{"_PRE",   METHOD_0ARGS,
680                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
681                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

683     {{"_PRL",   METHOD_0ARGS,
684                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
685                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

687     {{"_PRS",   METHOD_0ARGS,
688                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

690     /*
691      * For _PRT, many BIOSs reverse the 3rd and 4th Package elements (Source
692      * and SourceIndex). This bug is so prevalent that there is code in the
693      * ACPICA Resource Manager to detect this and switch them back. For now,
694      * do not allow and issue a warning. To allow this and eliminate the
695      * warning, add the ACPI_RTYPE_REFERENCE type to the 4th element (index 3)
696      * in the statement below.
697      */
698     {{"_PRT",   METHOD_0ARGS,
699                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
700                     PACKAGE_INFO (ACPI_PTYPE2_FIXED, 4, ACPI_RTYPE_INTEGER, ACPI
701                                   ACPI_RTYPE_INTEGER | ACPI_RTYPE_REFERENCE, ACP

703     {{"_PRW",   METHOD_0ARGS,
704                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
705                     PACKAGE_INFO (ACPI_PTYPE1_OPTION, 2, ACPI_RTYPE_INTEGER | AC
706                                   ACPI_RTYPE_INTEGER, ACPI_RTYPE_REFERENCE, 0),

708     {{"_PS0",   METHOD_0ARGS,
709                 METHOD_NO_RETURN_VALUE}},

711     {{"_PS1",   METHOD_0ARGS,
712                 METHOD_NO_RETURN_VALUE}},

714     {{"_PS2",   METHOD_0ARGS,
715                 METHOD_NO_RETURN_VALUE}},

717     {{"_PS3",   METHOD_0ARGS,
718                 METHOD_NO_RETURN_VALUE}},

720     {{"_PSC",   METHOD_0ARGS,
```



```
721                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

723     {{"_PSD",   METHOD_0ARGS,
724                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
725                     PACKAGE_INFO (ACPI_PTYPE2_COUNT, ACPI_RTYPE_INTEGER, 0,0,0,0

727     {{"_PSE",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
728                 METHOD_NO_RETURN_VALUE}},

730     {{"_PSL",   METHOD_0ARGS,
731                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
732                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

734     {{"_PSR",   METHOD_0ARGS,
735                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

737     {{"_PSS",   METHOD_0ARGS,
738                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
739                     PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_INTEGER, 6,0,0,0),

741     {{"_PSV",   METHOD_0ARGS,
742                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

744     {{"_PSW",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
745                 METHOD_NO_RETURN_VALUE}},

747     {{"_PTC",   METHOD_0ARGS,
748                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (2 Buf) *
749                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_BUFFER, 2,0,0,0)

751     {{"_PTP",   METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
752                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

754     {{"_PTS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
755                 METHOD_NO_RETURN_VALUE}},

757     {{"_PUR",   METHOD_0ARGS,
758                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (2 Int) *
759                     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 2,0,0,0

761     {{"_PXM",   METHOD_0ARGS,
762                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

764     {{"_REG",   METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
765                 METHOD_NO_RETURN_VALUE}},

767     {{"_REV",   METHOD_0ARGS,
768                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

770     {{"_RMV",   METHOD_0ARGS,
771                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

773     {{"_ROM",   METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
774                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

776     {{"_RTV",   METHOD_0ARGS,
777                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

779     /*
780      * For _S0_ through _S5_, the ACPI spec defines a return Package
781      * containing 1 Integer, but most DSDTs have it wrong - 2,3, or 4 integers.
782      * Allow this by making the objects "Variable-length length", but all elemen
783      * must be Integers.
784      */
785     {{"_S0_",   METHOD_0ARGS,
786                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
```

```
787                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
788
789     {{"_S1_",   METHOD_0ARGS,
790                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
791                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
792
793     {{"_S2_",   METHOD_0ARGS,
794                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
795                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
796
797     {{"_S3_",   METHOD_0ARGS,
798                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
799                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
800
801     {{"_S4_",   METHOD_0ARGS,
802                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
803                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
804
805     {{"_S5_",   METHOD_0ARGS,
806                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (1 Int) *
807                     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_INTEGER, 1,0,0,0),
808
809     {{"_S1D",   METHOD_0ARGS,
810                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
811
812     {{"_S2D",   METHOD_0ARGS,
813                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
814
815     {{"_S3D",   METHOD_0ARGS,
816                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
817
818     {{"_S4D",   METHOD_0ARGS,
819                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
820
821     {{"_S0W",   METHOD_0ARGS,
822                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
823
824     {{"_S1W",   METHOD_0ARGS,
825                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
826
827     {{"_S2W",   METHOD_0ARGS,
828                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
829
830     {{"_S3W",   METHOD_0ARGS,
831                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
832
833     {{"_S4W",   METHOD_0ARGS,
834                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
835
836     {{"_SBS",   METHOD_0ARGS,
837                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
838
839     {{"_SCP",   METHOD_1ARGS (ACPI_TYPE_INTEGER) | ARG_COUNT_IS_MINIMUM,
840                 METHOD_NO_RETURN_VALUE}},   /* Acpi 1.0 allowed 1 integer arg. A
841
842     {{"_SDD",   METHOD_1ARGS (ACPI_TYPE_BUFFER),
843                 METHOD_NO_RETURN_VALUE}},
844
845     {{"_SEG",   METHOD_0ARGS,
846                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
847
848     {{"_SHL",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
849                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
850
851     {{"_SLI",   METHOD_0ARGS,
852                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
```



```
853
854     {{"_SPD",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
855                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
856
857     {{"_SRS",   METHOD_1ARGS (ACPI_TYPE_BUFFER),
858                 METHOD_NO_RETURN_VALUE}},
859
860     {{"_SRT",   METHOD_1ARGS (ACPI_TYPE_BUFFER),
861                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
862
863     {{"_SRV",   METHOD_0ARGS,
864                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}, /* See IPMI spec */
865
866     {{"_SST",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
867                 METHOD_NO_RETURN_VALUE}},
868
869     {{"_STA",   METHOD_0ARGS,
870                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
871
872     {{"_STM",   METHOD_3ARGS (ACPI_TYPE_BUFFER, ACPI_TYPE_BUFFER, ACPI_TYPE_BUFF
873                 METHOD_NO_RETURN_VALUE}},
874
875     {{"_STP",   METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
876                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
877
878     {{"_STR",   METHOD_0ARGS,
879                 METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
880
881     {{"_STV",   METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
882                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
883
884     {{"_SUB",   METHOD_0ARGS,
885                 METHOD_RETURNS (ACPI_RTYPE_STRING)}},
886
887     {{"_SUN",   METHOD_0ARGS,
888                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
889
890     {{"_SWS",   METHOD_0ARGS,
891                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
892
893     {{"_TC1",   METHOD_0ARGS,
894                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
895
896     {{"_TC2",   METHOD_0ARGS,
897                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
898
899     {{"_TDL",   METHOD_0ARGS,
900                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
901
902     {{"_TIP",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
903                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
904
905     {{"_TIV",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
906                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
907
908     {{"_TMP",   METHOD_0ARGS,
909                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
910
911     {{"_TPC",   METHOD_0ARGS,
912                 METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
913
914     {{"_TPT",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
915                 METHOD_NO_RETURN_VALUE}},
916
917     {{"_TRT",   METHOD_0ARGS,
918                 METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
```

```
                    PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_REFERENCE, 2, ACPI_RTY

    {{"_TSD",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
                    PACKAGE_INFO (ACPI_PTYPE2_COUNT,ACPI_RTYPE_INTEGER, 5,0,0,0)

    {{"_TSP",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_TSS",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
                    PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_INTEGER, 5,0,0,0),

    {{"_TST",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_TTS",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
                METHOD_NO_RETURN_VALUE}},

    {{"_TZD",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
                    PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

    {{"_TZM",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_REFERENCE)}},

    {{"_TZP",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_UID",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING)}},

    {{"_UPC",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (4 Int) *
                    PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 4,0,0,0

    {{"_UPD",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_UPP",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    {{"_VPO",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

    /* Acpi 1.0 defined _WAK with no return value. Later, it was changed to retu

    {{"_WAK",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
                METHOD_RETURNS (ACPI_RTYPE_NONE | ACPI_RTYPE_INTEGER | ACPI_RTYP
                    PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 2,0,0,0

    /* _WDG/_WED are MS extensions defined by "Windows Instrumentation" */

    {{"_WDG",   METHOD_0ARGS,
                METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

    {{"_WED",   METHOD_1ARGS (ACPI_TYPE_INTEGER),
                METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RT

    PACKAGE_INFO (0,0,0,0,0,0) /* Table terminator */
};
#else
extern const ACPI_PREDEFINED_INFO       AcpiGbl_PredefinedMethods[];
#endif
```



```
#if (defined ACPI_CREATE_RESOURCE_TABLE && defined ACPI_APPLICATION)
/******************************************************************************
 *
 * Predefined names for use in Resource Descriptors. These names do not
 * appear in the global Predefined Name table (since these names never
 * appear in actual AML byte code, only in the original ASL)
 *
 * Note: Used by iASL compiler and AcpiHelp utility only.
 *
 *****************************************************************************/

const ACPI_PREDEFINED_INFO      AcpiGbl_ResourceNames[] =
{
    {{"_ADR",   WIDTH_16 | WIDTH_64,            0}},
    {{"_ALN",   WIDTH_8 | WIDTH_16 | WIDTH_32,  0}},
    {{"_ASI",   WIDTH_8,                        0}},
    {{"_ASZ",   WIDTH_8,                        0}},
    {{"_ATT",   WIDTH_64,                       0}},
    {{"_BAS",   WIDTH_16 | WIDTH_32,            0}},
    {{"_BM_",   WIDTH_1,                        0}},
    {{"_DBT",   WIDTH_16,                       0}},  /* Acpi 5.0 */
    {{"_DEC",   WIDTH_1,                        0}},
    {{"_DMA",   WIDTH_8,                        0}},
    {{"_DPL",   WIDTH_1,                        0}},  /* Acpi 5.0 */
    {{"_DRS",   WIDTH_16,                       0}},  /* Acpi 5.0 */
    {{"_END",   WIDTH_1,                        0}},  /* Acpi 5.0 */
    {{"_FLC",   WIDTH_2,                        0}},  /* Acpi 5.0 */
    {{"_GRA",   WIDTH_ADDRESS,                  0}},
    {{"_HE_",   WIDTH_1,                        0}},
    {{"_INT",   WIDTH_16 | WIDTH_32,            0}},
    {{"_IOR",   WIDTH_2,                        0}},  /* Acpi 5.0 */
    {{"_LEN",   WIDTH_8 | WIDTH_ADDRESS,        0}},
    {{"_LIN",   WIDTH_8,                        0}},  /* Acpi 5.0 */
    {{"_LL_",   WIDTH_1,                        0}},
    {{"_MAF",   WIDTH_1,                        0}},
    {{"_MAX",   WIDTH_ADDRESS,                  0}},
    {{"_MEM",   WIDTH_2,                        0}},
    {{"_MIF",   WIDTH_1,                        0}},
    {{"_MIN",   WIDTH_ADDRESS,                  0}},
    {{"_MOD",   WIDTH_1,                        0}},  /* Acpi 5.0 */
    {{"_MTP",   WIDTH_2,                        0}},
    {{"_PAR",   WIDTH_8,                        0}},  /* Acpi 5.0 */
    {{"_PHA",   WIDTH_1,                        0}},  /* Acpi 5.0 */
    {{"_PIN",   WIDTH_16,                       0}},  /* Acpi 5.0 */
    {{"_PPI",   WIDTH_8,                        0}},  /* Acpi 5.0 */
    {{"_POL",   WIDTH_1 | WIDTH_2,              0}},  /* Acpi 5.0 */
    {{"_RBO",   WIDTH_8,                        0}},
    {{"_RBW",   WIDTH_8,                        0}},
    {{"_RNG",   WIDTH_1,                        0}},
    {{"_RT_",   WIDTH_8,                        0}},  /* Acpi 3.0 */
    {{"_RW_",   WIDTH_1,                        0}},
    {{"_RXL",   WIDTH_16,                       0}},  /* Acpi 5.0 */
    {{"_SHR",   WIDTH_2,                        0}},
    {{"_SIZ",   WIDTH_2,                        0}},
    {{"_SLV",   WIDTH_1,                        0}},  /* Acpi 5.0 */
    {{"_SPE",   WIDTH_32,                       0}},  /* Acpi 5.0 */
    {{"_STB",   WIDTH_2,                        0}},  /* Acpi 5.0 */
    {{"_TRA",   WIDTH_ADDRESS,                  0}},
    {{"_TRS",   WIDTH_1,                        0}},
    {{"_TSF",   WIDTH_8,                        0}},  /* Acpi 3.0 */
    {{"_TTP",   WIDTH_1,                        0}},
    {{"_TXL",   WIDTH_16,                       0}},  /* Acpi 5.0 */
    {{"_TYP",   WIDTH_2 | WIDTH_16,             0}},
    {{"_VEN",   VARIABLE_DATA,                  0}},  /* Acpi 5.0 */
    PACKAGE_INFO (0,0,0,0,0,0) /* Table terminator */
};
```

```
1052 static const ACPI_PREDEFINED_INFO      AcpiGbl_ScopeNames[] = {
1053     {{"_GPE",      0,      0}},
1054     {{"_PR_",      0,      0}},
1055     {{"_SB_",      0,      0}},
1056     {{"_SI_",      0,      0}},
1057     {{"_TZ_",      0,      0}},
1058     PACKAGE_INFO (0,0,0,0,0,0) /* Table terminator */
1059 };
1060 #else
1061 extern const ACPI_PREDEFINED_INFO      AcpiGbl_ResourceNames[];
1062 #endif

1064 #endif
```

```
**********************************************************
   12788 Thu Dec 26 13:49:54 2013
new/usr/src/common/acpica/include/acresrc.h
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acresrc.h - Resource Manager function prototypes
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACRESRC_H__
#define __ACRESRC_H__

/* Need the AML resource descriptor structs */

#include "amlresrc.h"


/*
 * If possible, pack the following structures to byte alignment, since we
 * don't care about performance for debug output. Two cases where we cannot
 * pack the structures:
 *
 * 1) Hardware does not support misaligned memory transfers
 * 2) Compiler does not support pointers within packed structures
```



```
 */
#if (!defined(ACPI_MISALIGNMENT_NOT_SUPPORTED) && !defined(ACPI_PACKED_POINTERS_
#pragma pack(1)
#endif

/*
 * Individual entry for the resource conversion tables
 */
typedef const struct acpi_rsconvert_info
{
    UINT8                   Opcode;
    UINT8                   ResourceOffset;
    UINT8                   AmlOffset;
    UINT8                   Value;

} ACPI_RSCONVERT_INFO;

/* Resource conversion opcodes */

typedef enum
{
    ACPI_RSC_INITGET        = 0,
    ACPI_RSC_INITSET,
    ACPI_RSC_FLAGINIT,
    ACPI_RSC_1BITFLAG,
    ACPI_RSC_2BITFLAG,
    ACPI_RSC_3BITFLAG,
    ACPI_RSC_ADDRESS,
    ACPI_RSC_BITMASK,
    ACPI_RSC_BITMASK16,
    ACPI_RSC_COUNT,
    ACPI_RSC_COUNT16,
    ACPI_RSC_COUNT_GPIO_PIN,
    ACPI_RSC_COUNT_GPIO_RES,
    ACPI_RSC_COUNT_GPIO_VEN,
    ACPI_RSC_COUNT_SERIAL_RES,
    ACPI_RSC_COUNT_SERIAL_VEN,
    ACPI_RSC_DATA8,
    ACPI_RSC_EXIT_EQ,
    ACPI_RSC_EXIT_LE,
    ACPI_RSC_EXIT_NE,
    ACPI_RSC_LENGTH,
    ACPI_RSC_MOVE_GPIO_PIN,
    ACPI_RSC_MOVE_GPIO_RES,
    ACPI_RSC_MOVE_SERIAL_RES,
    ACPI_RSC_MOVE_SERIAL_VEN,
    ACPI_RSC_MOVE8,
    ACPI_RSC_MOVE16,
    ACPI_RSC_MOVE32,
    ACPI_RSC_MOVE64,
    ACPI_RSC_SET8,
    ACPI_RSC_SOURCE,
    ACPI_RSC_SOURCEX
#define ACPI_RSC_INITGET                0
#define ACPI_RSC_INITSET                1
#define ACPI_RSC_FLAGINIT               2
#define ACPI_RSC_1BITFLAG               3
#define ACPI_RSC_2BITFLAG               4
#define ACPI_RSC_COUNT                  5
#define ACPI_RSC_COUNT16                6
#define ACPI_RSC_LENGTH                 7
#define ACPI_RSC_MOVE8                  8
#define ACPI_RSC_MOVE16                 9
#define ACPI_RSC_MOVE32                 10
#define ACPI_RSC_MOVE64                 11
#define ACPI_RSC_SET8                   12
```

```
 91 #define ACPI_RSC_DATA8                  13
 92 #define ACPI_RSC_ADDRESS                14
 93 #define ACPI_RSC_SOURCE                 15
 94 #define ACPI_RSC_SOURCEX                16
 95 #define ACPI_RSC_BITMASK                17
 96 #define ACPI_RSC_BITMASK16              18
 97 #define ACPI_RSC_EXIT_NE                19
 98 #define ACPI_RSC_EXIT_LE                20
 99 #define ACPI_RSC_EXIT_EQ                21

113 } ACPI_RSCONVERT_OPCODES;

115 /* Resource Conversion sub-opcodes */

117 #define ACPI_RSC_COMPARE_AML_LENGTH     0
118 #define ACPI_RSC_COMPARE_VALUE          1

120 #define ACPI_RSC_TABLE_SIZE(d)          (sizeof (d) / sizeof (ACPI_RSCONVERT_INF

122 #define ACPI_RS_OFFSET(f)               (UINT8) ACPI_OFFSET (ACPI_RESOURCE,f)
123 #define AML_OFFSET(f)                   (UINT8) ACPI_OFFSET (AML_RESOURCE,f)


126 /*
127  * Individual entry for the resource dump tables
128  */
129 typedef const struct acpi_rsdump_info
130 {
131     UINT8                   Opcode;
132     UINT8                   Offset;
133     char                    *Name;
134     const char              **Pointer;

136 } ACPI_RSDUMP_INFO;

138 /* Values for the Opcode field above */

140 typedef enum
141 {
142     ACPI_RSD_TITLE          = 0,
143     ACPI_RSD_1BITFLAG,
144     ACPI_RSD_2BITFLAG,
145     ACPI_RSD_3BITFLAG,
146     ACPI_RSD_ADDRESS,
147     ACPI_RSD_DWORDLIST,
148     ACPI_RSD_LITERAL,
149     ACPI_RSD_LONGLIST,
150     ACPI_RSD_SHORTLIST,
151     ACPI_RSD_SHORTLISTX,
152     ACPI_RSD_SOURCE,
153     ACPI_RSD_STRING,
154     ACPI_RSD_UINT8,
155     ACPI_RSD_UINT16,
156     ACPI_RSD_UINT32,
157     ACPI_RSD_UINT64,
158     ACPI_RSD_WORDLIST
123 #define ACPI_RSD_TITLE                  0
124 #define ACPI_RSD_LITERAL                1
125 #define ACPI_RSD_STRING                 2
126 #define ACPI_RSD_UINT8                  3
127 #define ACPI_RSD_UINT16                 4
128 #define ACPI_RSD_UINT32                 5
129 #define ACPI_RSD_UINT64                 6
130 #define ACPI_RSD_1BITFLAG               7
131 #define ACPI_RSD_2BITFLAG               8
132 #define ACPI_RSD_SHORTLIST              9
```



```
133 #define ACPI_RSD_LONGLIST               10
134 #define ACPI_RSD_DWORDLIST              11
135 #define ACPI_RSD_ADDRESS                12
136 #define ACPI_RSD_SOURCE                 13

160 } ACPI_RSDUMP_OPCODES;

162 /* restore default alignment */

164 #pragma pack()


167 /* Resource tables indexed by internal resource type */

169 extern const UINT8              AcpiGbl_AmlResourceSizes[];
170 extern const UINT8              AcpiGbl_AmlResourceSerialBusSizes[];
171 extern ACPI_RSCONVERT_INFO      *AcpiGbl_SetResourceDispatch[];

173 /* Resource tables indexed by raw AML resource descriptor type */

175 extern const UINT8              AcpiGbl_ResourceStructSizes[];
176 extern const UINT8              AcpiGbl_ResourceStructSerialBusSizes[];
177 extern ACPI_RSCONVERT_INFO      *AcpiGbl_GetResourceDispatch[];

179 extern ACPI_RSCONVERT_INFO      *AcpiGbl_ConvertResourceSerialBusDispatch[];

181 typedef struct acpi_vendor_walk_info
182 {
183     ACPI_VENDOR_UUID        *Uuid;
184     ACPI_BUFFER             *Buffer;
185     ACPI_STATUS             Status;

187 } ACPI_VENDOR_WALK_INFO;


190 /*
191  * rscreate
192  */
193 ACPI_STATUS
194 AcpiRsCreateResourceList (
195     ACPI_OPERAND_OBJECT     *AmlBuffer,
196     ACPI_BUFFER             *OutputBuffer);

198 ACPI_STATUS
199 AcpiRsCreateAmlResources (
200     ACPI_BUFFER             *ResourceList,
173     ACPI_RESOURCE           *LinkedListBuffer,
201     ACPI_BUFFER             *OutputBuffer);

203 ACPI_STATUS
204 AcpiRsCreatePciRoutingTable (
205     ACPI_OPERAND_OBJECT     *PackageObject,
206     ACPI_BUFFER             *OutputBuffer);


209 /*
210  * rsutils
211  */
212 ACPI_STATUS
213 AcpiRsGetPrtMethodData (
214     ACPI_NAMESPACE_NODE     *Node,
215     ACPI_BUFFER             *RetBuffer);

217 ACPI_STATUS
218 AcpiRsGetCrsMethodData (
219     ACPI_NAMESPACE_NODE     *Node,
```

```
220     ACPI_BUFFER             *RetBuffer);

222 ACPI_STATUS
223 AcpiRsGetPrsMethodData (
224     ACPI_NAMESPACE_NODE     *Node,
225     ACPI_BUFFER             *RetBuffer);

227 ACPI_STATUS
228 AcpiRsGetMethodData (
229     ACPI_HANDLE             Handle,
230     char                    *Path,
231     ACPI_BUFFER             *RetBuffer);

233 ACPI_STATUS
234 AcpiRsSetSrsMethodData (
235     ACPI_NAMESPACE_NODE     *Node,
236     ACPI_BUFFER             *RetBuffer);

238 ACPI_STATUS
239 AcpiRsGetAeiMethodData (
240     ACPI_NAMESPACE_NODE     *Node,
241     ACPI_BUFFER             *RetBuffer);

243 /*
244  * rscalc
245  */
246 ACPI_STATUS
247 AcpiRsGetListLength (
248     UINT8                   *AmlBuffer,
249     UINT32                  AmlBufferLength,
250     ACPI_SIZE               *SizeNeeded);

252 ACPI_STATUS
253 AcpiRsGetAmlLength (
254     ACPI_RESOURCE           *ResourceList,
255     ACPI_SIZE               ResourceListSize,
223     ACPI_RESOURCE           *LinkedListBuffer,
256     ACPI_SIZE               *SizeNeeded);

258 ACPI_STATUS
259 AcpiRsGetPciRoutingTableLength (
260     ACPI_OPERAND_OBJECT     *PackageObject,
261     ACPI_SIZE               *BufferSizeNeeded);

263 ACPI_STATUS
264 AcpiRsConvertAmlToResources (
265     UINT8                   *Aml,
266     UINT32                  Length,
267     UINT32                  Offset,
268     UINT8                   ResourceIndex,
269     void                    **Context);
237     void                    *Context);

271 ACPI_STATUS
272 AcpiRsConvertResourcesToAml (
273     ACPI_RESOURCE           *Resource,
274     ACPI_SIZE               AmlSizeNeeded,
275     UINT8                   *OutputBuffer);


278 /*
279  * rsaddr
280  */
281 void
282 AcpiRsSetAddressCommon (
283     AML_RESOURCE            *Aml,
```



```
284     ACPI_RESOURCE           *Resource);

286 BOOLEAN
287 AcpiRsGetAddressCommon (
288     ACPI_RESOURCE           *Resource,
289     AML_RESOURCE            *Aml);


292 /*
293  * rsmisc
294  */
295 ACPI_STATUS
296 AcpiRsConvertAmlToResource (
297     ACPI_RESOURCE           *Resource,
298     AML_RESOURCE            *Aml,
299     ACPI_RSCONVERT_INFO     *Info);

301 ACPI_STATUS
302 AcpiRsConvertResourceToAml (
303     ACPI_RESOURCE           *Resource,
304     AML_RESOURCE            *Aml,
305     ACPI_RSCONVERT_INFO     *Info);


308 /*
309  * rsutils
310  */
311 void
312 AcpiRsMoveData (
313     void                    *Destination,
314     void                    *Source,
315     UINT16                  ItemCount,
316     UINT8                   MoveType);

318 UINT8
319 AcpiRsDecodeBitmask (
320     UINT16                  Mask,
321     UINT8                   *List);

323 UINT16
324 AcpiRsEncodeBitmask (
325     UINT8                   *List,
326     UINT8                   Count);

328 ACPI_RS_LENGTH
329 AcpiRsGetResourceSource (
330     ACPI_RS_LENGTH          ResourceLength,
331     ACPI_RS_LENGTH          MinimumLength,
332     ACPI_RESOURCE_SOURCE    *ResourceSource,
333     AML_RESOURCE            *Aml,
334     char                    *StringPtr);

336 ACPI_RSDESC_SIZE
337 AcpiRsSetResourceSource (
338     AML_RESOURCE            *Aml,
339     ACPI_RS_LENGTH          MinimumLength,
340     ACPI_RESOURCE_SOURCE    *ResourceSource);

342 void
343 AcpiRsSetResourceHeader (
344     UINT8                   DescriptorType,
345     ACPI_RSDESC_SIZE        TotalLength,
346     AML_RESOURCE            *Aml);

348 void
349 AcpiRsSetResourceLength (
```

```
    ACPI_RSDESC_SIZE        TotalLength,
    AML_RESOURCE            *Aml);


/*
 * rsdump
 */
void
AcpiRsDumpResourceList (
    ACPI_RESOURCE           *Resource);

void
AcpiRsDumpIrqList (
    UINT8                   *RouteTable);


/*
 * Resource conversion tables
 */
extern ACPI_RSCONVERT_INFO      AcpiRsConvertDma[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertEndDpf[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertIo[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertFixedIo[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertEndTag[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertMemory24[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertGenericReg[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertMemory32[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertFixedMemory32[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertAddress32[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertAddress16[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertExtIrq[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertAddress64[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertExtAddress64[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertGpio[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertFixedDma[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertI2cSerialBus[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertSpiSerialBus[];
extern ACPI_RSCONVERT_INFO      AcpiRsConvertUartSerialBus[];

/* These resources require separate get/set tables */

extern ACPI_RSCONVERT_INFO      AcpiRsGetIrq[];
extern ACPI_RSCONVERT_INFO      AcpiRsGetStartDpf[];
extern ACPI_RSCONVERT_INFO      AcpiRsGetVendorSmall[];
extern ACPI_RSCONVERT_INFO      AcpiRsGetVendorLarge[];

extern ACPI_RSCONVERT_INFO      AcpiRsSetIrq[];
extern ACPI_RSCONVERT_INFO      AcpiRsSetStartDpf[];
extern ACPI_RSCONVERT_INFO      AcpiRsSetVendor[];


#if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)
/*
 * rsinfo
 */
extern ACPI_RSDUMP_INFO         *AcpiGbl_DumpResourceDispatch[];
extern ACPI_RSDUMP_INFO         *AcpiGbl_DumpSerialBusDispatch[];

/*
 * rsdumpinfo
 * rsdump
 */
extern ACPI_RSDUMP_INFO         AcpiRsDumpIrq[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpPrt[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpDma[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpStartDpf[];
```



```
extern ACPI_RSDUMP_INFO         AcpiRsDumpEndDpf[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpIo[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpIoFlags[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpFixedIo[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpVendor[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpEndTag[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpMemory24[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpMemory32[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpMemoryFlags[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpFixedMemory32[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpAddress16[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpAddress32[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpAddress64[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpExtAddress64[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpExtIrq[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpGenericReg[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpGpio[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpFixedDma[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpCommonSerialBus[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpI2cSerialBus[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpSpiSerialBus[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpUartSerialBus[];
extern ACPI_RSDUMP_INFO         AcpiRsDumpGeneralFlags[];
#endif

#endif  /* __ACRESRC_H__ */
```

```
**********************************************************
   23494 Thu Dec 26 13:49:54 2013
new/usr/src/common/acpica/include/acrestyp.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acrestyp.h - Defines, types, and structures for resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACRESTYP_H__
#define __ACRESTYP_H__


/*
 * Definitions for Resource Attributes
 */
typedef UINT16                          ACPI_RS_LENGTH;    /* Resource Length fi
typedef UINT32                          ACPI_RSDESC_SIZE;  /* Max Resource Descr

/*
 * Memory Attributes
 */
#define ACPI_READ_ONLY_MEMORY           (UINT8) 0x00
#define ACPI_READ_WRITE_MEMORY          (UINT8) 0x01

#define ACPI_NON_CACHEABLE_MEMORY       (UINT8) 0x00
```



```
#define ACPI_CACHABLE_MEMORY            (UINT8) 0x01
#define ACPI_WRITE_COMBINING_MEMORY     (UINT8) 0x02
#define ACPI_PREFETCHABLE_MEMORY        (UINT8) 0x03

/*! [Begin] no source code translation */
/*
 * IO Attributes
 * The ISA IO ranges are:     n000-n0FFh,  n400-n4FFh, n800-n8FFh, nC00-nCFFh.
 * The non-ISA IO ranges are: n100-n3FFh,  n500-n7FFh, n900-nBFFh, nCD0-nFFFh.
 */
/*! [End] no source code translation !*/

#define ACPI_NON_ISA_ONLY_RANGES        (UINT8) 0x01
#define ACPI_ISA_ONLY_RANGES            (UINT8) 0x02
#define ACPI_ENTIRE_RANGE               (ACPI_NON_ISA_ONLY_RANGES | ACPI_ISA_ONL

/* Type of translation - 1=Sparse, 0=Dense */

#define ACPI_SPARSE_TRANSLATION         (UINT8) 0x01

/*
 * IO Port Descriptor Decode
 */
#define ACPI_DECODE_10                  (UINT8) 0x00    /* 10-bit IO address dec
#define ACPI_DECODE_16                  (UINT8) 0x01    /* 16-bit IO address dec

/*
 * Interrupt attributes - used in multiple descriptors
 */

/* Triggering */

#define ACPI_LEVEL_SENSITIVE            (UINT8) 0x00
#define ACPI_EDGE_SENSITIVE             (UINT8) 0x01

/* Polarity */

#define ACPI_ACTIVE_HIGH                (UINT8) 0x00
#define ACPI_ACTIVE_LOW                 (UINT8) 0x01
#define ACPI_ACTIVE_BOTH                (UINT8) 0x02

/* Sharing */

#define ACPI_EXCLUSIVE                  (UINT8) 0x00
#define ACPI_SHARED                     (UINT8) 0x01

/* Wake */

#define ACPI_NOT_WAKE_CAPABLE           (UINT8) 0x00
#define ACPI_WAKE_CAPABLE               (UINT8) 0x01

/*
 * DMA Attributes
 */
#define ACPI_COMPATIBILITY              (UINT8) 0x00
#define ACPI_TYPE_A                     (UINT8) 0x01
#define ACPI_TYPE_B                     (UINT8) 0x02
#define ACPI_TYPE_F                     (UINT8) 0x03

#define ACPI_NOT_BUS_MASTER             (UINT8) 0x00
#define ACPI_BUS_MASTER                 (UINT8) 0x01

#define ACPI_TRANSFER_8                 (UINT8) 0x00
#define ACPI_TRANSFER_8_16              (UINT8) 0x01
#define ACPI_TRANSFER_16                (UINT8) 0x02
```

```
127 /*
128  * Start Dependent Functions Priority definitions
129  */
130 #define ACPI_GOOD_CONFIGURATION         (UINT8) 0x00
131 #define ACPI_ACCEPTABLE_CONFIGURATION   (UINT8) 0x01
132 #define ACPI_SUB_OPTIMAL_CONFIGURATION  (UINT8) 0x02

134 /*
135  * 16, 32 and 64-bit Address Descriptor resource types
136  */
137 #define ACPI_MEMORY_RANGE               (UINT8) 0x00
138 #define ACPI_IO_RANGE                   (UINT8) 0x01
139 #define ACPI_BUS_NUMBER_RANGE           (UINT8) 0x02

141 #define ACPI_ADDRESS_NOT_FIXED          (UINT8) 0x00
142 #define ACPI_ADDRESS_FIXED              (UINT8) 0x01

144 #define ACPI_POS_DECODE                 (UINT8) 0x00
145 #define ACPI_SUB_DECODE                 (UINT8) 0x01

147 /* Producer/Consumer */

149 #define ACPI_PRODUCER                   (UINT8) 0x00
150 #define ACPI_CONSUMER                   (UINT8) 0x01


153 /*
154  * If possible, pack the following structures to byte alignment
155  */
156 #ifndef ACPI_MISALIGNMENT_NOT_SUPPORTED
157 #pragma pack(1)
158 #endif

160 /* UUID data structures for use in vendor-defined resource descriptors */

162 typedef struct acpi_uuid
163 {
164     UINT8                           Data[ACPI_UUID_LENGTH];
165 } ACPI_UUID;

167 typedef struct acpi_vendor_uuid
168 {
169     UINT8                           Subtype;
170     UINT8                           Data[ACPI_UUID_LENGTH];

172 } ACPI_VENDOR_UUID;

174 /*
175  * Structures used to describe device resources
176  */
177 typedef struct acpi_resource_irq
178 {
179     UINT8                           DescriptorLength;
180     UINT8                           Triggering;
181     UINT8                           Polarity;
182     UINT8                           Sharable;
183     UINT8                           WakeCapable;
184     UINT8                           InterruptCount;
185     UINT8                           Interrupts[1];

187 } ACPI_RESOURCE_IRQ;

189 typedef struct acpi_resource_dma
190 {
191     UINT8                           Type;
192     UINT8                           BusMaster;
```



```
193     UINT8                           Transfer;
194     UINT8                           ChannelCount;
195     UINT8                           Channels[1];

197 } ACPI_RESOURCE_DMA;

199 typedef struct acpi_resource_start_dependent
200 {
201     UINT8                           DescriptorLength;
202     UINT8                           CompatibilityPriority;
203     UINT8                           PerformanceRobustness;

205 } ACPI_RESOURCE_START_DEPENDENT;


208 /*
209  * The END_DEPENDENT_FUNCTIONS_RESOURCE struct is not
210  * needed because it has no fields
211  */


214 typedef struct acpi_resource_io
215 {
216     UINT8                           IoDecode;
217     UINT8                           Alignment;
218     UINT8                           AddressLength;
219     UINT16                          Minimum;
220     UINT16                          Maximum;

222 } ACPI_RESOURCE_IO;

224 typedef struct acpi_resource_fixed_io
225 {
226     UINT16                          Address;
227     UINT8                           AddressLength;

229 } ACPI_RESOURCE_FIXED_IO;

231 typedef struct acpi_resource_fixed_dma
232 {
233     UINT16                          RequestLines;
234     UINT16                          Channels;
235     UINT8                           Width;

237 } ACPI_RESOURCE_FIXED_DMA;

239 /* Values for Width field above */

241 #define ACPI_DMA_WIDTH8                         0
242 #define ACPI_DMA_WIDTH16                        1
243 #define ACPI_DMA_WIDTH32                        2
244 #define ACPI_DMA_WIDTH64                        3
245 #define ACPI_DMA_WIDTH128                       4
246 #define ACPI_DMA_WIDTH256                       5


249 typedef struct acpi_resource_vendor
250 {
251     UINT16                          ByteLength;
252     UINT8                           ByteData[1];

254 } ACPI_RESOURCE_VENDOR;

256 /* Vendor resource with UUID info (introduced in ACPI 3.0) */

258 typedef struct acpi_resource_vendor_typed
```

```
259 {
260     UINT16                          ByteLength;
261     UINT8                           UuidSubtype;
262     UINT8                           Uuid[ACPI_UUID_LENGTH];
263     UINT8                           ByteData[1];

265 } ACPI_RESOURCE_VENDOR_TYPED;

267 typedef struct acpi_resource_end_tag
268 {
269     UINT8                           Checksum;

271 } ACPI_RESOURCE_END_TAG;

273 typedef struct acpi_resource_memory24
274 {
275     UINT8                           WriteProtect;
276     UINT16                          Minimum;
277     UINT16                          Maximum;
278     UINT16                          Alignment;
279     UINT16                          AddressLength;

281 } ACPI_RESOURCE_MEMORY24;

283 typedef struct acpi_resource_memory32
284 {
285     UINT8                           WriteProtect;
286     UINT32                          Minimum;
287     UINT32                          Maximum;
288     UINT32                          Alignment;
289     UINT32                          AddressLength;

291 } ACPI_RESOURCE_MEMORY32;

293 typedef struct acpi_resource_fixed_memory32
294 {
295     UINT8                           WriteProtect;
296     UINT32                          Address;
297     UINT32                          AddressLength;

299 } ACPI_RESOURCE_FIXED_MEMORY32;

301 typedef struct acpi_memory_attribute
302 {
303     UINT8                           WriteProtect;
304     UINT8                           Caching;
305     UINT8                           RangeType;
306     UINT8                           Translation;

308 } ACPI_MEMORY_ATTRIBUTE;

310 typedef struct acpi_io_attribute
311 {
312     UINT8                           RangeType;
313     UINT8                           Translation;
314     UINT8                           TranslationType;
315     UINT8                           Reserved1;

317 } ACPI_IO_ATTRIBUTE;

319 typedef union acpi_resource_attribute
320 {
321     ACPI_MEMORY_ATTRIBUTE           Mem;
322     ACPI_IO_ATTRIBUTE               Io;

324     /* Used for the *WordSpace macros */
```



```
326     UINT8                           TypeSpecific;

328 } ACPI_RESOURCE_ATTRIBUTE;

330 typedef struct acpi_resource_source
331 {
332     UINT8                           Index;
333     UINT16                          StringLength;
334     char                            *StringPtr;

336 } ACPI_RESOURCE_SOURCE;

338 /* Fields common to all address descriptors, 16/32/64 bit */

340 #define ACPI_RESOURCE_ADDRESS_COMMON \
341     UINT8                           ResourceType; \
342     UINT8                           ProducerConsumer; \
343     UINT8                           Decode; \
344     UINT8                           MinAddressFixed; \
345     UINT8                           MaxAddressFixed; \
346     ACPI_RESOURCE_ATTRIBUTE         Info;

348 typedef struct acpi_resource_address
349 {
350     ACPI_RESOURCE_ADDRESS_COMMON

352 } ACPI_RESOURCE_ADDRESS;

354 typedef struct acpi_resource_address16
355 {
356     ACPI_RESOURCE_ADDRESS_COMMON
357     UINT16                          Granularity;
358     UINT16                          Minimum;
359     UINT16                          Maximum;
360     UINT16                          TranslationOffset;
361     UINT16                          AddressLength;
362     ACPI_RESOURCE_SOURCE            ResourceSource;

364 } ACPI_RESOURCE_ADDRESS16;

366 typedef struct acpi_resource_address32
367 {
368     ACPI_RESOURCE_ADDRESS_COMMON
369     UINT32                          Granularity;
370     UINT32                          Minimum;
371     UINT32                          Maximum;
372     UINT32                          TranslationOffset;
373     UINT32                          AddressLength;
374     ACPI_RESOURCE_SOURCE            ResourceSource;

376 } ACPI_RESOURCE_ADDRESS32;

378 typedef struct acpi_resource_address64
379 {
380     ACPI_RESOURCE_ADDRESS_COMMON
381     UINT64                          Granularity;
382     UINT64                          Minimum;
383     UINT64                          Maximum;
384     UINT64                          TranslationOffset;
385     UINT64                          AddressLength;
386     ACPI_RESOURCE_SOURCE            ResourceSource;

388 } ACPI_RESOURCE_ADDRESS64;

390 typedef struct acpi_resource_extended_address64
```

```
391 {
392     ACPI_RESOURCE_ADDRESS_COMMON
393     UINT8                           RevisionID;
394     UINT64                          Granularity;
395     UINT64                          Minimum;
396     UINT64                          Maximum;
397     UINT64                          TranslationOffset;
398     UINT64                          AddressLength;
399     UINT64                          TypeSpecific;

401 } ACPI_RESOURCE_EXTENDED_ADDRESS64;

403 typedef struct acpi_resource_extended_irq
404 {
405     UINT8                           ProducerConsumer;
406     UINT8                           Triggering;
407     UINT8                           Polarity;
408     UINT8                           Sharable;
409     UINT8                           WakeCapable;
410     UINT8                           InterruptCount;
411     ACPI_RESOURCE_SOURCE            ResourceSource;
412     UINT32                          Interrupts[1];

414 } ACPI_RESOURCE_EXTENDED_IRQ;

416 typedef struct acpi_resource_generic_register
417 {
418     UINT8                           SpaceId;
419     UINT8                           BitWidth;
420     UINT8                           BitOffset;
421     UINT8                           AccessSize;
422     UINT64                          Address;

424 } ACPI_RESOURCE_GENERIC_REGISTER;

426 typedef struct acpi_resource_gpio
427 {
428     UINT8                           RevisionId;
429     UINT8                           ConnectionType;
430     UINT8                           ProducerConsumer;   /* For values, see Produ
431     UINT8                           PinConfig;
432     UINT8                           Sharable;           /* For values, see Inter
433     UINT8                           WakeCapable;        /* For values, see Inter
434     UINT8                           IoRestriction;
435     UINT8                           Triggering;         /* For values, see Inter
436     UINT8                           Polarity;           /* For values, see Inter
437     UINT16                          DriveStrength;
438     UINT16                          DebounceTimeout;
439     UINT16                          PinTableLength;
440     UINT16                          VendorLength;
441     ACPI_RESOURCE_SOURCE            ResourceSource;
442     UINT16                          *PinTable;
443     UINT8                           *VendorData;

445 } ACPI_RESOURCE_GPIO;

447 /* Values for GPIO ConnectionType field above */

449 #define ACPI_RESOURCE_GPIO_TYPE_INT             0
450 #define ACPI_RESOURCE_GPIO_TYPE_IO              1

452 /* Values for PinConfig field above */

454 #define ACPI_PIN_CONFIG_DEFAULT                 0
455 #define ACPI_PIN_CONFIG_PULLUP                  1
456 #define ACPI_PIN_CONFIG_PULLDOWN                2
```



```
457 #define ACPI_PIN_CONFIG_NOPULL                  3

459 /* Values for IoRestriction field above */

461 #define ACPI_IO_RESTRICT_NONE                   0
462 #define ACPI_IO_RESTRICT_INPUT                  1
463 #define ACPI_IO_RESTRICT_OUTPUT                 2
464 #define ACPI_IO_RESTRICT_NONE_PRESERVE          3


467 /* Common structure for I2C, SPI, and UART serial descriptors */

469 #define ACPI_RESOURCE_SERIAL_COMMON \
470     UINT8                           RevisionId; \
471     UINT8                           Type; \
472     UINT8                           ProducerConsumer;    /* For values, see Prod
473     UINT8                           SlaveMode; \
474     UINT8                           TypeRevisionId; \
475     UINT16                          TypeDataLength; \
476     UINT16                          VendorLength; \
477     ACPI_RESOURCE_SOURCE            ResourceSource; \
478     UINT8                           *VendorData;

480 typedef struct acpi_resource_common_serialbus
481 {
482     ACPI_RESOURCE_SERIAL_COMMON

484 } ACPI_RESOURCE_COMMON_SERIALBUS;

486 /* Values for the Type field above */

488 #define ACPI_RESOURCE_SERIAL_TYPE_I2C           1
489 #define ACPI_RESOURCE_SERIAL_TYPE_SPI           2
490 #define ACPI_RESOURCE_SERIAL_TYPE_UART          3

492 /* Values for SlaveMode field above */

494 #define ACPI_CONTROLLER_INITIATED               0
495 #define ACPI_DEVICE_INITIATED                   1


498 typedef struct acpi_resource_i2c_serialbus
499 {
500     ACPI_RESOURCE_SERIAL_COMMON
501     UINT8                           AccessMode;
502     UINT16                          SlaveAddress;
503     UINT32                          ConnectionSpeed;

505 } ACPI_RESOURCE_I2C_SERIALBUS;

507 /* Values for AccessMode field above */

509 #define ACPI_I2C_7BIT_MODE                      0
510 #define ACPI_I2C_10BIT_MODE                     1


513 typedef struct acpi_resource_spi_serialbus
514 {
515     ACPI_RESOURCE_SERIAL_COMMON
516     UINT8                           WireMode;
517     UINT8                           DevicePolarity;
518     UINT8                           DataBitLength;
519     UINT8                           ClockPhase;
520     UINT8                           ClockPolarity;
521     UINT16                          DeviceSelection;
522     UINT32                          ConnectionSpeed;
```

```
} ACPI_RESOURCE_SPI_SERIALBUS;

/* Values for WireMode field above */

#define ACPI_SPI_4WIRE_MODE                     0
#define ACPI_SPI_3WIRE_MODE                     1

/* Values for DevicePolarity field above */

#define ACPI_SPI_ACTIVE_LOW                     0
#define ACPI_SPI_ACTIVE_HIGH                    1

/* Values for ClockPhase field above */

#define ACPI_SPI_FIRST_PHASE                    0
#define ACPI_SPI_SECOND_PHASE                   1

/* Values for ClockPolarity field above */

#define ACPI_SPI_START_LOW                      0
#define ACPI_SPI_START_HIGH                     1


typedef struct acpi_resource_uart_serialbus
{
    ACPI_RESOURCE_SERIAL_COMMON
    UINT8                           Endian;
    UINT8                           DataBits;
    UINT8                           StopBits;
    UINT8                           FlowControl;
    UINT8                           Parity;
    UINT8                           LinesEnabled;
    UINT16                          RxFifoSize;
    UINT16                          TxFifoSize;
    UINT32                          DefaultBaudRate;

} ACPI_RESOURCE_UART_SERIALBUS;

/* Values for Endian field above */

#define ACPI_UART_LITTLE_ENDIAN                 0
#define ACPI_UART_BIG_ENDIAN                    1

/* Values for DataBits field above */

#define ACPI_UART_5_DATA_BITS                   0
#define ACPI_UART_6_DATA_BITS                   1
#define ACPI_UART_7_DATA_BITS                   2
#define ACPI_UART_8_DATA_BITS                   3
#define ACPI_UART_9_DATA_BITS                   4

/* Values for StopBits field above */

#define ACPI_UART_NO_STOP_BITS                  0
#define ACPI_UART_1_STOP_BIT                    1
#define ACPI_UART_1P5_STOP_BITS                 2
#define ACPI_UART_2_STOP_BITS                   3

/* Values for FlowControl field above */

#define ACPI_UART_FLOW_CONTROL_NONE             0
#define ACPI_UART_FLOW_CONTROL_HW               1
#define ACPI_UART_FLOW_CONTROL_XON_XOFF         2

/* Values for Parity field above */
```



```
#define ACPI_UART_PARITY_NONE                   0
#define ACPI_UART_PARITY_EVEN                   1
#define ACPI_UART_PARITY_ODD                    2
#define ACPI_UART_PARITY_MARK                   3
#define ACPI_UART_PARITY_SPACE                  4

/* Values for LinesEnabled bitfield above */

#define ACPI_UART_CARRIER_DETECT                (1<<2)
#define ACPI_UART_RING_INDICATOR                (1<<3)
#define ACPI_UART_DATA_SET_READY                (1<<4)
#define ACPI_UART_DATA_TERMINAL_READY           (1<<5)
#define ACPI_UART_CLEAR_TO_SEND                 (1<<6)
#define ACPI_UART_REQUEST_TO_SEND               (1<<7)


/* ACPI_RESOURCE_TYPEs */

#define ACPI_RESOURCE_TYPE_IRQ                  0
#define ACPI_RESOURCE_TYPE_DMA                  1
#define ACPI_RESOURCE_TYPE_START_DEPENDENT      2
#define ACPI_RESOURCE_TYPE_END_DEPENDENT        3
#define ACPI_RESOURCE_TYPE_IO                   4
#define ACPI_RESOURCE_TYPE_FIXED_IO             5
#define ACPI_RESOURCE_TYPE_VENDOR               6
#define ACPI_RESOURCE_TYPE_END_TAG              7
#define ACPI_RESOURCE_TYPE_MEMORY24             8
#define ACPI_RESOURCE_TYPE_MEMORY32             9
#define ACPI_RESOURCE_TYPE_FIXED_MEMORY32       10
#define ACPI_RESOURCE_TYPE_ADDRESS16            11
#define ACPI_RESOURCE_TYPE_ADDRESS32            12
#define ACPI_RESOURCE_TYPE_ADDRESS64            13
#define ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64   14  /* ACPI 3.0 */
#define ACPI_RESOURCE_TYPE_EXTENDED_IRQ         15
#define ACPI_RESOURCE_TYPE_GENERIC_REGISTER     16
#define ACPI_RESOURCE_TYPE_GPIO                 17  /* ACPI 5.0 */
#define ACPI_RESOURCE_TYPE_FIXED_DMA            18  /* ACPI 5.0 */
#define ACPI_RESOURCE_TYPE_SERIAL_BUS           19  /* ACPI 5.0 */
#define ACPI_RESOURCE_TYPE_MAX                  19

/* Master union for resource descriptors */

typedef union acpi_resource_data
{
    ACPI_RESOURCE_IRQ                       Irq;
    ACPI_RESOURCE_DMA                       Dma;
    ACPI_RESOURCE_START_DEPENDENT           StartDpf;
    ACPI_RESOURCE_IO                        Io;
    ACPI_RESOURCE_FIXED_IO                  FixedIo;
    ACPI_RESOURCE_FIXED_DMA                 FixedDma;
    ACPI_RESOURCE_VENDOR                    Vendor;
    ACPI_RESOURCE_VENDOR_TYPED              VendorTyped;
    ACPI_RESOURCE_END_TAG                   EndTag;
    ACPI_RESOURCE_MEMORY24                  Memory24;
    ACPI_RESOURCE_MEMORY32                  Memory32;
    ACPI_RESOURCE_FIXED_MEMORY32            FixedMemory32;
    ACPI_RESOURCE_ADDRESS16                 Address16;
    ACPI_RESOURCE_ADDRESS32                 Address32;
    ACPI_RESOURCE_ADDRESS64                 Address64;
    ACPI_RESOURCE_EXTENDED_ADDRESS64        ExtAddress64;
    ACPI_RESOURCE_EXTENDED_IRQ              ExtendedIrq;
    ACPI_RESOURCE_GENERIC_REGISTER          GenericReg;
    ACPI_RESOURCE_GPIO                      Gpio;
    ACPI_RESOURCE_I2C_SERIALBUS             I2cSerialBus;
    ACPI_RESOURCE_SPI_SERIALBUS             SpiSerialBus;
```

```
655     ACPI_RESOURCE_UART_SERIALBUS            UartSerialBus;
656     ACPI_RESOURCE_COMMON_SERIALBUS          CommonSerialBus;

658     /* Common fields */

660     ACPI_RESOURCE_ADDRESS                   Address;        /* Common 16/32/64 a

662 } ACPI_RESOURCE_DATA;


665 /* Common resource header */

667 typedef struct acpi_resource
668 {
669     UINT32                          Type;
670     UINT32                          Length;
671     ACPI_RESOURCE_DATA              Data;

673 } ACPI_RESOURCE;

675 /* restore default alignment */

677 #pragma pack()


680 #define ACPI_RS_SIZE_NO_DATA                8       /* Id + Length fields */
681 #define ACPI_RS_SIZE_MIN                    (UINT32) ACPI_ROUND_UP_TO_NATIVE_WOR
682 #define ACPI_RS_SIZE(Type)                  (UINT32) (ACPI_RS_SIZE_NO_DATA + siz

684 /* Macro for walking resource templates with multiple descriptors */

686 #define ACPI_NEXT_RESOURCE(Res) \
687     ACPI_ADD_PTR (ACPI_RESOURCE, (Res), (Res)->Length)


690 typedef struct acpi_pci_routing_table
691 {
692     UINT32                          Length;
693     UINT32                          Pin;
694     UINT64                          Address;        /* here for 64-bit alignment
695     UINT32                          SourceIndex;
696     char                            Source[4];      /* pad to 64 bits so sizeof(

698 } ACPI_PCI_ROUTING_TABLE;

700 #endif /* __ACRESTYP_H__ */
```

```
**********************************************************
   11317 Thu Dec 26 13:49:55 2013
new/usr/src/common/acpica/include/acstruct.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acstruct.h - Internal structs
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACSTRUCT_H__
#define __ACSTRUCT_H__

/* acpisrc:StructDefs -- for acpisrc conversion */

/******************************************************************************
 *
 * Tree walking typedefs and structs
 *
 *****************************************************************************/


/*
 * Walk state - current state of a parse tree walk. Used for both a leisurely
 * stroll through the tree (for whatever reason), and for control method
 * execution.
 */
```



```
 */
#define ACPI_NEXT_OP_DOWNWARD       1
#define ACPI_NEXT_OP_UPWARD         2

/*
 * Groups of definitions for WalkType used for different implementations of
 * walkers (never simultaneously) - flags for interpreter:
 */
#define ACPI_WALK_NON_METHOD        0
#define ACPI_WALK_METHOD            0x01
#define ACPI_WALK_METHOD_RESTART    0x02

/* Flags for iASL compiler only */

#define ACPI_WALK_CONST_REQUIRED    0x10
#define ACPI_WALK_CONST_OPTIONAL    0x20


typedef struct acpi_walk_state
{
    struct acpi_walk_state          *Next;              /* Next WalkState in lis
    UINT8                           DescriptorType;     /* To differentiate vari
    UINT8                           WalkType;
    UINT16                          Opcode;             /* Current AML opcode */
    UINT8                           NextOpInfo;         /* Info about NextOp */
    UINT8                           NumOperands;        /* Stack pointer for Ope
    UINT8                           OperandIndex;       /* Index into operand st
    ACPI_OWNER_ID                   OwnerId;            /* Owner of objects crea
    BOOLEAN                         LastPredicate;      /* Result of last predic
    UINT8                           CurrentResult;
    UINT8                           ReturnUsed;
    UINT8                           ScopeDepth;
    UINT8                           PassNumber;         /* Parse pass during tab
    UINT8                           ResultSize;         /* Total elements for th
    UINT8                           ResultCount;        /* Current number of occ
    UINT32                          AmlOffset;
    UINT32                          ArgTypes;
    UINT32                          MethodBreakpoint;   /* For single stepping *
    UINT32                          UserBreakpoint;     /* User AML breakpoint *
    UINT32                          ParseFlags;

    ACPI_PARSE_STATE                ParserState;        /* Current state of pars
    UINT32                          PrevArgTypes;
    UINT32                          ArgCount;           /* push for fixed or var

    struct acpi_namespace_node      Arguments[ACPI_METHOD_NUM_ARGS];        /* C
    struct acpi_namespace_node      LocalVariables[ACPI_METHOD_NUM_LOCALS]; /* C
    union acpi_operand_object       *Operands[ACPI_OBJ_NUM_OPERANDS + 1];   /* O
    union acpi_operand_object       **Params;

    UINT8                           *AmlLastWhile;
    union acpi_operand_object       **CallerReturnDesc;
    ACPI_GENERIC_STATE              *ControlState;      /* List of control state
    struct acpi_namespace_node      *DeferredNode;      /* Used when executing d
    union acpi_operand_object       *ImplicitReturnObj;
    struct acpi_namespace_node      *MethodCallNode;    /* Called method Node*/
    ACPI_PARSE_OBJECT               *MethodCallOp;      /* MethodCall Op if runn
    union acpi_operand_object       *MethodDesc;        /* Method descriptor if
    struct acpi_namespace_node      *MethodNode;        /* Method node if runnin
    ACPI_PARSE_OBJECT               *Op;                /* Current parser op */
    const ACPI_OPCODE_INFO          *OpInfo;            /* Info on current opcod
    ACPI_PARSE_OBJECT               *Origin;            /* Start of walk [Obsole
    union acpi_operand_object       *ResultObj;
    ACPI_GENERIC_STATE              *Results;           /* Stack of accumulated
    union acpi_operand_object       *ReturnDesc;        /* Return object, if any
    ACPI_GENERIC_STATE              *ScopeInfo;         /* Stack of nested scope
```

```
 126     ACPI_PARSE_OBJECT               *PrevOp;            /* Last op that was proc
 127     ACPI_PARSE_OBJECT               *NextOp;            /* next op to be process
 128     ACPI_THREAD_STATE               *Thread;
 129     ACPI_PARSE_DOWNWARDS            DescendingCallback;
 130     ACPI_PARSE_UPWARDS              AscendingCallback;

 132 } ACPI_WALK_STATE;
______unchanged_portion_omitted_


 197 /*
 198  * Structure used to pass object evaluation information and parameters.
 198  * Structure used to pass object evaluation parameters.
 199  * Purpose is to reduce CPU stack use.
 200  */
 201 typedef struct acpi_evaluate_info
 202 {
 203     /* The first 3 elements are passed by the caller to AcpiNsEvaluate */
 203     ACPI_NAMESPACE_NODE             *PrefixNode;
 204     char                            *Pathname;
 205     ACPI_OPERAND_OBJECT             *ObjDesc;
 206     ACPI_OPERAND_OBJECT             **Parameters;
 207     ACPI_NAMESPACE_NODE             *ResolvedNode;
 208     ACPI_OPERAND_OBJECT             *ReturnObject;
 209     UINT8                           ParamCount;
 210     UINT8                           PassNumber;
 211     UINT8                           ReturnObjectType;
 212     UINT8                           Flags;

 205     ACPI_NAMESPACE_NODE             *PrefixNode;        /* Input: starting node
 206     char                            *RelativePathname;  /* Input: path relative
 207     ACPI_OPERAND_OBJECT             **Parameters;       /* Input: argument list

 209     ACPI_NAMESPACE_NODE             *Node;              /* Resolved node (Prefix
 210     ACPI_OPERAND_OBJECT             *ObjDesc;           /* Object attached to th
 211     char                            *FullPathname;      /* Full pathname of the

 213     const ACPI_PREDEFINED_INFO      *Predefined;        /* Used if Node is a pre
 214     ACPI_OPERAND_OBJECT             *ReturnObject;      /* Object returned from
 215     union acpi_operand_object       *ParentPackage;     /* Used if return object

 217     UINT32                          ReturnFlags;        /* Used for return value
 218     UINT32                          ReturnBtype;        /* Bitmapped type of the
 219     UINT16                          ParamCount;         /* Count of the input ar
 220     UINT8                           PassNumber;         /* Parser pass number */
 221     UINT8                           ReturnObjectType;   /* Object type of the re
 222     UINT8                           NodeFlags;          /* Same as Node->Flags *
 223     UINT8                           Flags;              /* General flags */

 225 } ACPI_EVALUATE_INFO;

 227 /* Values for Flags above */

 229 #define ACPI_IGNORE_RETURN_VALUE    1

 231 /* Defines for ReturnFlags field above */

 233 #define ACPI_OBJECT_REPAIRED        1
 234 #define ACPI_OBJECT_WRAPPED         2


 237 /* Info used by AcpiNsInitializeDevices */

 239 typedef struct acpi_device_walk_info
 240 {
 241     ACPI_TABLE_DESC                 *TableDesc;
```



```
 242     ACPI_EVALUATE_INFO              *EvaluateInfo;
 243     UINT32                          DeviceCount;
 244     UINT32                          Num_STA;
 245     UINT32                          Num_INI;

 247 } ACPI_DEVICE_WALK_INFO;
______unchanged_portion_omitted_
```

```
**********************************************************
    4978 Thu Dec 26 13:49:55 2013
new/usr/src/common/acpica/include/actables.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Name: actables.h - ACPI table management
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACTABLES_H__
#define __ACTABLES_H__


ACPI_STATUS
AcpiAllocateRootTable (
    UINT32                  InitialTableCount);

/*
 * tbxfroot - Root pointer utilities
 */
ACPI_STATUS
AcpiTbValidateRsdp (
    ACPI_TABLE_RSDP         *Rsdp);

UINT8 *
```



```
AcpiTbScanMemoryForRsdp (
    UINT8                   *StartAddress,
    UINT32                  Length);


/*
 * tbfadt - FADT parse/convert/validate
 */
void
AcpiTbParseFadt (
    UINT32                  TableIndex);

void
AcpiTbCreateLocalFadt (
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Length);


/*
 * tbfind - find ACPI table
 */
ACPI_STATUS
AcpiTbFindTable (
    char                    *Signature,
    char                    *OemId,
    char                    *OemTableId,
    UINT32                  *TableIndex);


/*
 * tbinstal - Table removal and deletion
 */
ACPI_STATUS
AcpiTbResizeRootTableList (
    void);

ACPI_STATUS
AcpiTbVerifyTable (
    ACPI_TABLE_DESC         *TableDesc);

ACPI_TABLE_HEADER *
AcpiTbTableOverride (
    ACPI_TABLE_HEADER       *TableHeader,
    ACPI_TABLE_DESC         *TableDesc);

ACPI_STATUS
AcpiTbAddTable (
    ACPI_TABLE_DESC         *TableDesc,
    UINT32                  *TableIndex);

ACPI_STATUS
AcpiTbStoreTable (
    ACPI_PHYSICAL_ADDRESS   Address,
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Length,
    UINT8                   Flags,
    UINT32                  *TableIndex);

void
AcpiTbDeleteTable (
    ACPI_TABLE_DESC        *TableDesc);

void
AcpiTbTerminate (
    void);
```

```
126 ACPI_STATUS
127 AcpiTbDeleteNamespaceByOwner (
128     UINT32                  TableIndex);

130 ACPI_STATUS
131 AcpiTbAllocateOwnerId (
132     UINT32                  TableIndex);

134 ACPI_STATUS
135 AcpiTbReleaseOwnerId (
136     UINT32                  TableIndex);

138 ACPI_STATUS
139 AcpiTbGetOwnerId (
140     UINT32                  TableIndex,
141     ACPI_OWNER_ID           *OwnerId);

143 BOOLEAN
144 AcpiTbIsTableLoaded (
145     UINT32                  TableIndex);

147 void
148 AcpiTbSetTableLoadedFlag (
149     UINT32                  TableIndex,
150     BOOLEAN                 IsLoaded);


153 /*
154  * tbutils - table manager utilities
155  */
156 ACPI_STATUS
157 AcpiTbInitializeFacs (
158     void);

160 BOOLEAN
161 AcpiTbTablesLoaded (
162     void);

164 void
165 AcpiTbPrintTableHeader(
166     ACPI_PHYSICAL_ADDRESS   Address,
167     ACPI_TABLE_HEADER       *Header);

169 UINT8
170 AcpiTbChecksum (
171     UINT8                   *Buffer,
172     UINT32                  Length);

174 ACPI_STATUS
175 AcpiTbVerifyChecksum (
176     ACPI_TABLE_HEADER       *Table,
177     UINT32                  Length);

179 void
180 AcpiTbCheckDsdtHeader (
181     void);

183 ACPI_TABLE_HEADER *
184 AcpiTbCopyDsdt (
185     UINT32                  TableIndex);

187 void
188 AcpiTbInstallTable (
189     ACPI_PHYSICAL_ADDRESS   Address,
190     char                    *Signature,
191     UINT32                  TableIndex);
```



```
193 ACPI_STATUS
194 AcpiTbParseRootTable (
195     ACPI_PHYSICAL_ADDRESS   RsdpAddress);

197 #endif /* __ACTABLES_H__ */
```

```
**********************************************************
   22475 Thu Dec 26 13:49:55 2013
new/usr/src/common/acpica/include/actbl.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: actbl.h - Basic ACPI Table Definitions
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACTBL_H__
  45 #define __ACTBL_H__


  48 /******************************************************************************
  49  *
  50  * Fundamental ACPI tables
  51  *
  52  * This file contains definitions for the ACPI tables that are directly consumed
  53  * by ACPICA. All other tables are consumed by the OS-dependent ACPI-related
  54  * device drivers and other OS support code.
  55  *
  56  * The RSDP and FACS do not use the common ACPI table header. All other ACPI
  57  * tables use the header.
  58  *
```



```
  59  *****************************************************************************/


  62 /*
  63  * Values for description table header signatures for tables defined in this
  64  * file. Useful because they make it more difficult to inadvertently type in
  65  * the wrong signature.
  66  */
  67 #define ACPI_SIG_DSDT           "DSDT"      /* Differentiated System Description
  68 #define ACPI_SIG_FADT           "FACP"      /* Fixed ACPI Description Table */
  69 #define ACPI_SIG_FACS           "FACS"      /* Firmware ACPI Control Structure *
  70 #define ACPI_SIG_PSDT           "PSDT"      /* Persistent System Description Tab
  71 #define ACPI_SIG_RSDP           "RSD PTR "  /* Root System Description Pointer *
  72 #define ACPI_SIG_RSDT           "RSDT"      /* Root System Description Table */
  73 #define ACPI_SIG_XSDT           "XSDT"      /* Extended  System Description Tabl
  74 #define ACPI_SIG_SSDT           "SSDT"      /* Secondary System Description Tabl
  75 #define ACPI_RSDP_NAME          "RSDP"      /* Short name for RSDP, not signatur


  78 /*
  79  * All tables and structures must be byte-packed to match the ACPI
  80  * specification, since the tables are provided by the system BIOS
  81  */
  82 #pragma pack(1)

  84 /*
  85  * Note: C bitfields are not used for this reason:
  86  *
  87  * "Bitfields are great and easy to read, but unfortunately the C language
  88  * does not specify the layout of bitfields in memory, which means they are
  89  * essentially useless for dealing with packed data in on-disk formats or
  90  * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
  91  * this decision was a design error in C. Ritchie could have picked an order
  92  * and stuck with it." Norman Ramsey.
  93  * See http://stackoverflow.com/a/1053662/41661
  85  * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
  86  * This is the only type that is even remotely portable. Anything else is not
  87  * portable, so do not use any other bitfield types.
  94  */


  97 /******************************************************************************
  98  *
  99  * Master ACPI Table Header. This common header is used by all ACPI tables
 100  * except the RSDP and FACS.
 101  *
 102  *****************************************************************************/

 104 typedef struct acpi_table_header
 105 {
 106     char                    Signature[ACPI_NAME_SIZE];          /* ASCII table s
 107     UINT32                  Length;                             /* Length of tab
 108     UINT8                   Revision;                           /* ACPI Specific
 102     UINT8                   Revision;                           /* ACPI Specific
 109     UINT8                   Checksum;                           /* To make sum o
 110     char                    OemId[ACPI_OEM_ID_SIZE];            /* ASCII OEM ide
 111     char                    OemTableId[ACPI_OEM_TABLE_ID_SIZE]; /* ASCII OEM tab
 112     UINT32                  OemRevision;                        /* OEM revision
 113     char                    AslCompilerId[ACPI_NAME_SIZE];      /* ASCII ASL com
 114     UINT32                  AslCompilerRevision;                /* ASL compiler

 116 } ACPI_TABLE_HEADER;


 119 /******************************************************************************
 120  *
```

```
 121  * GAS - Generic Address Structure (ACPI 2.0+)
 122  *
 123  * Note: Since this structure is used in the ACPI tables, it is byte aligned.
 124  * If misaligned access is not supported by the hardware, accesses to the
 118  * If misaligned access is not supported by the hardware, accesses to the
 125  * 64-bit Address field must be performed with care.
 126  *
 127  ******************************************************************************/

 129 typedef struct acpi_generic_address
 130 {
 131     UINT8                   SpaceId;                /* Address space where struc
 132     UINT8                   BitWidth;               /* Size in bits of given reg
 133     UINT8                   BitOffset;              /* Bit offset within the reg
 134     UINT8                   AccessWidth;            /* Minimum Access size (ACPI
 135     UINT64                  Address;                /* 64-bit address of struct

 137 } ACPI_GENERIC_ADDRESS;
_____unchanged_portion_omitted_

 206 #define ACPI_RSDT_ENTRY_SIZE        (sizeof (UINT32))
 207 #define ACPI_XSDT_ENTRY_SIZE        (sizeof (UINT64))


 210 /******************************************************************************
 211  *
 212  * FACS - Firmware ACPI Control Structure (FACS)
 213  *
 214  ******************************************************************************/

 216 typedef struct acpi_table_facs
 217 {
 218     char                    Signature[4];           /* ASCII table signature */
 219     UINT32                  Length;                 /* Length of structure, in b
 220     UINT32                  HardwareSignature;      /* Hardware configuration si
 221     UINT32                  FirmwareWakingVector;   /* 32-bit physical address o
 222     UINT32                  GlobalLock;             /* Global Lock for shared ha
 223     UINT32                  Flags;
 224     UINT64                  XFirmwareWakingVector;  /* 64-bit version of the Fir
 225     UINT8                   Version;                /* Version of this table (AC
 226     UINT8                   Reserved[3];            /* Reserved, must be zero */
 227     UINT32                  OspmFlags;              /* Flags to be set by OSPM (
 228     UINT8                   Reserved1[24];          /* Reserved, must be zero */

 230 } ACPI_TABLE_FACS;

 232 /* Masks for GlobalLock flag field above */

 234 #define ACPI_GLOCK_PENDING          (1)             /* 00: Pending global lock o
 235 #define ACPI_GLOCK_OWNED            (1<<1)          /* 01: Global lock is owned

 237 /* Masks for Flags field above  */

 239 #define ACPI_FACS_S4_BIOS_PRESENT   (1)             /* 00: S4BIOS support is pre
 240 #define ACPI_FACS_64BIT_WAKE        (1<<1)          /* 01: 64-bit wake vector su

 242 /* Masks for OspmFlags field above */

 244 #define ACPI_FACS_64BIT_ENVIRONMENT (1)             /* 00: 64-bit wake environme


 247 /******************************************************************************
 248  *
 249  * FADT - Fixed ACPI Description Table (Signature "FACP")
 250  *        Version 4
 251  *
```



```
 252  ******************************************************************************/

 254 /* Fields common to all versions of the FADT */

 256 typedef struct acpi_table_fadt
 257 {
 258     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
 259     UINT32                  Facs;               /* 32-bit physical address of FA
 260     UINT32                  Dsdt;               /* 32-bit physical address of DS
 261     UINT8                   Model;              /* System Interrupt Model (ACPI
 262     UINT8                   PreferredProfile;   /* Conveys preferred power manag
 263     UINT16                  SciInterrupt;       /* System vector of SCI interrup
 264     UINT32                  SmiCommand;         /* 32-bit Port address of SMI co
 265     UINT8                   AcpiEnable;         /* Value to write to SMI_CMD to
 266     UINT8                   AcpiDisable;        /* Value to write to SMI_CMD to
 267     UINT8                   S4BiosRequest;      /* Value to write to SMI_CMD to
 256     UINT8                   AcpiEnable;         /* Value to write to smi_cmd to
 257     UINT8                   AcpiDisable;        /* Value to write to smi_cmd to
 258     UINT8                   S4BiosRequest;      /* Value to write to SMI CMD to
 268     UINT8                   PstateControl;      /* Processor performance state c
 269     UINT32                  Pm1aEventBlock;     /* 32-bit port address of Power
 270     UINT32                  Pm1bEventBlock;     /* 32-bit port address of Power
 271     UINT32                  Pm1aControlBlock;   /* 32-bit port address of Power
 272     UINT32                  Pm1bControlBlock;   /* 32-bit port address of Power
 273     UINT32                  Pm2ControlBlock;    /* 32-bit port address of Power
 274     UINT32                  PmTimerBlock;       /* 32-bit port address of Power
 275     UINT32                  Gpe0Block;          /* 32-bit port address of Genera
 276     UINT32                  Gpe1Block;          /* 32-bit port address of Genera
 260     UINT32                  Pm1aEventBlock;     /* 32-bit Port address of Power
 261     UINT32                  Pm1bEventBlock;     /* 32-bit Port address of Power
 262     UINT32                  Pm1aControlBlock;   /* 32-bit Port address of Power
 263     UINT32                  Pm1bControlBlock;   /* 32-bit Port address of Power
 264     UINT32                  Pm2ControlBlock;    /* 32-bit Port address of Power
 265     UINT32                  PmTimerBlock;       /* 32-bit Port address of Power
 266     UINT32                  Gpe0Block;          /* 32-bit Port address of Genera
 267     UINT32                  Gpe1Block;          /* 32-bit Port address of Genera
 277     UINT8                   Pm1EventLength;     /* Byte Length of ports at Pm1xE
 278     UINT8                   Pm1ControlLength;   /* Byte Length of ports at Pm1xC
 279     UINT8                   Pm2ControlLength;   /* Byte Length of ports at Pm2Co
 280     UINT8                   PmTimerLength;      /* Byte Length of ports at PmTim
 281     UINT8                   Gpe0BlockLength;    /* Byte Length of ports at Gpe0B
 282     UINT8                   Gpe1BlockLength;    /* Byte Length of ports at Gpe1B
 283     UINT8                   Gpe1Base;           /* Offset in GPE number space wh
 284     UINT8                   CstControl;         /* Support for the _CST object a
 275     UINT8                   CstControl;         /* Support for the _CST object a
 285     UINT16                  C2Latency;          /* Worst case HW latency to ente
 286     UINT16                  C3Latency;          /* Worst case HW latency to ente
 287     UINT16                  FlushSize;          /* Processor memory cache line w
 278     UINT16                  FlushSize;          /* Processor's memory cache line
 288     UINT16                  FlushStride;        /* Number of flush strides that
 289     UINT8                   DutyOffset;         /* Processor duty cycle index in
 280     UINT8                   DutyOffset;         /* Processor duty cycle index in
 290     UINT8                   DutyWidth;          /* Processor duty cycle value bi
 291     UINT8                   DayAlarm;           /* Index to day-of-month alarm i
 292     UINT8                   MonthAlarm;         /* Index to month-of-year alarm
 293     UINT8                   Century;            /* Index to century in RTC CMOS
 294     UINT16                  BootFlags;          /* IA-PC Boot Architecture Flags
 295     UINT8                   Reserved;           /* Reserved, must be zero */
 296     UINT32                  Flags;              /* Miscellaneous flag bits (see
 297     ACPI_GENERIC_ADDRESS    ResetRegister;      /* 64-bit address of the Reset r
 298     UINT8                   ResetValue;         /* Value to write to the ResetRe
 299     UINT8                   Reserved4[3];       /* Reserved, must be zero */
 300     UINT64                  XFacs;              /* 64-bit physical address of FA
 301     UINT64                  XDsdt;              /* 64-bit physical address of DS
 302     ACPI_GENERIC_ADDRESS    XPm1aEventBlock;    /* 64-bit Extended Power Mgt 1a
 303     ACPI_GENERIC_ADDRESS    XPm1bEventBlock;    /* 64-bit Extended Power Mgt 1b
```

```
    ACPI_GENERIC_ADDRESS    XPm1aControlBlock;  /* 64-bit Extended Power Mgt 1a
    ACPI_GENERIC_ADDRESS    XPm1bControlBlock;  /* 64-bit Extended Power Mgt 1b
    ACPI_GENERIC_ADDRESS    XPm2ControlBlock;   /* 64-bit Extended Power Mgt 2 C
    ACPI_GENERIC_ADDRESS    XPmTimerBlock;      /* 64-bit Extended Power Mgt Tim
    ACPI_GENERIC_ADDRESS    XGpe0Block;         /* 64-bit Extended General Purpo
    ACPI_GENERIC_ADDRESS    XGpe1Block;         /* 64-bit Extended General Purpo
    ACPI_GENERIC_ADDRESS    SleepControl;       /* 64-bit Sleep Control register
    ACPI_GENERIC_ADDRESS    SleepStatus;        /* 64-bit Sleep Status register

} ACPI_TABLE_FADT;


/* Masks for FADT Boot Architecture Flags (BootFlags) [Vx]=Introduced in this FA
/* Masks for FADT Boot Architecture Flags (BootFlags) */

#define ACPI_FADT_LEGACY_DEVICES    (1)         /* 00: [V2] System has LPC or IS
#define ACPI_FADT_8042              (1<<1)      /* 01: [V3] System has an 8042 c
#define ACPI_FADT_NO_VGA            (1<<2)      /* 02: [V4] It is not safe to pr
#define ACPI_FADT_NO_MSI            (1<<3)      /* 03: [V4] Message Signaled Int
#define ACPI_FADT_NO_ASPM           (1<<4)      /* 04: [V4] PCIe ASPM control mu
#define ACPI_FADT_NO_CMOS_RTC       (1<<5)      /* 05: [V5] No CMOS real-time cl

/* Masks for FADT flags */

#define ACPI_FADT_WBINVD            (1)         /* 00: [V1] The WBINVD instructi
#define ACPI_FADT_WBINVD_FLUSH      (1<<1)      /* 01: [V1] WBINVD flushes but d
#define ACPI_FADT_WBINVD            (1)         /* 00: [V1] The wbinvd instructi
#define ACPI_FADT_WBINVD_FLUSH      (1<<1)      /* 01: [V1] wbinvd flushes but d
#define ACPI_FADT_C1_SUPPORTED      (1<<2)      /* 02: [V1] All processors suppo
#define ACPI_FADT_C2_MP_SUPPORTED   (1<<3)      /* 03: [V1] C2 state works on MP
#define ACPI_FADT_POWER_BUTTON      (1<<4)      /* 04: [V1] Power button is hand
#define ACPI_FADT_SLEEP_BUTTON      (1<<5)      /* 05: [V1] Sleep button is hand
#define ACPI_FADT_FIXED_RTC         (1<<6)      /* 06: [V1] RTC wakeup status is
#define ACPI_FADT_FIXED_RTC         (1<<6)      /* 06: [V1] RTC wakeup status no
#define ACPI_FADT_S4_RTC_WAKE       (1<<7)      /* 07: [V1] RTC alarm can wake s
#define ACPI_FADT_32BIT_TIMER       (1<<8)      /* 08: [V1] ACPI timer width is
#define ACPI_FADT_DOCKING_SUPPORTED (1<<9)      /* 09: [V1] Docking supported */
#define ACPI_FADT_RESET_REGISTER    (1<<10)     /* 10: [V2] System reset via the
#define ACPI_FADT_SEALED_CASE       (1<<11)     /* 11: [V3] No internal expansio
#define ACPI_FADT_HEADLESS          (1<<12)     /* 12: [V3] No local video capab
#define ACPI_FADT_SLEEP_TYPE        (1<<13)     /* 13: [V3] Must execute native
#define ACPI_FADT_PCI_EXPRESS_WAKE  (1<<14)     /* 14: [V4] System supports PCIE
#define ACPI_FADT_PLATFORM_CLOCK    (1<<15)     /* 15: [V4] OSPM should use plat
#define ACPI_FADT_S4_RTC_VALID      (1<<16)     /* 16: [V4] Contents of RTC_STS
#define ACPI_FADT_REMOTE_POWER_ON   (1<<17)     /* 17: [V4] System is compatible
#define ACPI_FADT_APIC_CLUSTER      (1<<18)     /* 18: [V4] All local APICs must
#define ACPI_FADT_APIC_PHYSICAL     (1<<19)     /* 19: [V4] All local xAPICs mus
#define ACPI_FADT_HW_REDUCED        (1<<20)     /* 20: [V5] ACPI hardware is not
#define ACPI_FADT_LOW_POWER_S0      (1<<21)     /* 21: [V5] S0 power savings are


/* Values for PreferredProfile (Preferred Power Management Profiles) */
/* Values for PreferredProfile (Prefered Power Management Profiles) */

enum AcpiPreferredPmProfiles
enum AcpiPreferedPmProfiles
{
    PM_UNSPECIFIED          = 0,
    PM_DESKTOP              = 1,
    PM_MOBILE               = 2,
    PM_WORKSTATION          = 3,
    PM_ENTERPRISE_SERVER    = 4,
    PM_SOHO_SERVER          = 5,
    PM_APPLIANCE_PC         = 6,
    PM_PERFORMANCE_SERVER   = 7,
    PM_TABLET               = 8
```



```
    PM_APPLIANCE_PC         = 6
};

/* Values for SleepStatus and SleepControl registers (V5 FADT) */

#define ACPI_X_WAKE_STATUS          0x80
#define ACPI_X_SLEEP_TYPE_MASK      0x1C
#define ACPI_X_SLEEP_TYPE_POSITION  0x02
#define ACPI_X_SLEEP_ENABLE         0x20


/* Reset to default packing */

#pragma pack()


/*
 * Internal table-related structures
 */
typedef union acpi_name_union
{
    UINT32                          Integer;
    char                            Ascii[4];

} ACPI_NAME_UNION;


/* Internal ACPI Table Descriptor. One per ACPI table. */

typedef struct acpi_table_desc
{
    ACPI_PHYSICAL_ADDRESS           Address;
    ACPI_TABLE_HEADER               *Pointer;
    UINT32                          Length;     /* Length fixed at 32 bits (fixe
    UINT32                          Length;     /* Length fixed at 32 bits */
    ACPI_NAME_UNION                 Signature;
    ACPI_OWNER_ID                   OwnerId;
    UINT8                           Flags;

} ACPI_TABLE_DESC;

/* Masks for Flags field above */

#define ACPI_TABLE_ORIGIN_UNKNOWN       (0)
#define ACPI_TABLE_ORIGIN_MAPPED        (1)
#define ACPI_TABLE_ORIGIN_ALLOCATED     (2)
#define ACPI_TABLE_ORIGIN_OVERRIDE      (4)
#define ACPI_TABLE_ORIGIN_MASK          (7)
#define ACPI_TABLE_IS_LOADED            (8)


/*
 * Get the remaining ACPI tables
 */
#include "actbl1.h"
#include "actbl2.h"
#include "actbl3.h"

/* Macros used to generate offsets to specific table fields */

#define ACPI_FADT_OFFSET(f)             (UINT16) ACPI_OFFSET (ACPI_TABLE_FADT, f
#define ACPI_FADT_OFFSET(f)             (UINT8) ACPI_OFFSET (ACPI_TABLE_FADT, f)

/*
 * Sizes of the various flavors of FADT. We need to look closely
 * at the FADT length because the version number essentially tells
```

```
 427  * us nothing because of many BIOS bugs where the version does not
 428  * match the expected length. In other words, the length of the
 429  * FADT is the bottom line as to what the version really is.
 430  *
 431  * For reference, the values below are as follows:
 432  *      FADT V1  size: 0x074
 433  *      FADT V2  size: 0x084
 434  *      FADT V3  size: 0x0F4
 435  *      FADT V4  size: 0x0F4
 436  *      FADT V5  size: 0x10C
 408  *      FADT V1  size: 0x74
 409  *      FADT V2  size: 0x84
 410  *      FADT V3+ size: 0xF4
 437  */
 438 #define ACPI_FADT_V1_SIZE       (UINT32) (ACPI_FADT_OFFSET (Flags) + 4)
 439 #define ACPI_FADT_V2_SIZE       (UINT32) (ACPI_FADT_OFFSET (Reserved4[0]) + 3)
 440 #define ACPI_FADT_V3_SIZE       (UINT32) (ACPI_FADT_OFFSET (SleepControl))
 441 #define ACPI_FADT_V5_SIZE       (UINT32) (sizeof (ACPI_TABLE_FADT))
 414 #define ACPI_FADT_V3_SIZE       (UINT32) (sizeof (ACPI_TABLE_FADT))

 443 #endif /* __ACTBL_H__ */
```

```
**********************************************************
   36382 Thu Dec 26 13:49:56 2013
new/usr/src/common/acpica/include/actbl1.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: actbl1.h - Additional ACPI table definitions
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACTBL1_H__
  45 #define __ACTBL1_H__


  48 /******************************************************************************
  49  *
  50  * Additional ACPI Tables (1)
  51  *
  52  * These tables are not consumed directly by the ACPICA subsystem, but are
  53  * included here to support device drivers and the AML disassembler.
  54  *
  55  * The tables in this file are fully defined within the ACPI specification.
  56  *
  57  *****************************************************************************/
```



```
  60 /*
  61  * Values for description table header signatures for tables defined in this
  62  * file. Useful because they make it more difficult to inadvertently type in
  63  * the wrong signature.
  64  */
  65 #define ACPI_SIG_BERT           "BERT"      /* Boot Error Record Table */
  66 #define ACPI_SIG_CPEP           "CPEP"      /* Corrected Platform Error Polling
  67 #define ACPI_SIG_ECDT           "ECDT"      /* Embedded Controller Boot Resource
  68 #define ACPI_SIG_EINJ           "EINJ"      /* Error Injection table */
  69 #define ACPI_SIG_ERST           "ERST"      /* Error Record Serialization Table
  70 #define ACPI_SIG_HEST           "HEST"      /* Hardware Error Source Table */
  71 #define ACPI_SIG_MADT           "APIC"      /* Multiple APIC Description Table *
  72 #define ACPI_SIG_MSCT           "MSCT"      /* Maximum System Characteristics Ta
  73 #define ACPI_SIG_SBST           "SBST"      /* Smart Battery Specification Table
  74 #define ACPI_SIG_SLIT           "SLIT"      /* System Locality Distance Informat
  75 #define ACPI_SIG_SRAT           "SRAT"      /* System Resource Affinity Table */


  78 /*
  79  * All tables must be byte-packed to match the ACPI specification, since
  80  * the tables are provided by the system BIOS.
  81  */
  82 #pragma pack(1)

  84 /*
  85  * Note: C bitfields are not used for this reason:
  86  *
  87  * "Bitfields are great and easy to read, but unfortunately the C language
  88  * does not specify the layout of bitfields in memory, which means they are
  89  * essentially useless for dealing with packed data in on-disk formats or
  90  * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
  91  * this decision was a design error in C. Ritchie could have picked an order
  92  * and stuck with it." Norman Ramsey.
  93  * See http://stackoverflow.com/a/1053662/41661
  85  * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
  86  * This is the only type that is even remotely portable. Anything else is not
  87  * portable, so do not use any other bitfield types.
  94  */


  97 /******************************************************************************
  98  *
  99  * Common subtable headers
 100  *
 101  *****************************************************************************/

 103 /* Generic subtable header (used in MADT, SRAT, etc.) */

 105 typedef struct acpi_subtable_header
 106 {
 107     UINT8                   Type;
 108     UINT8                   Length;

 110 } ACPI_SUBTABLE_HEADER;
_____unchanged_portion_omitted_


 128 /******************************************************************************
 129  *
 130  * BERT - Boot Error Record Table (ACPI 4.0)
 131  *        Version 1
 132  *
 133  *****************************************************************************/

 135 typedef struct acpi_table_bert
```

```
136 {
137     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
138     UINT32                  RegionLength;       /* Length of the boot error regi
139     UINT64                  Address;            /* Physical address of the error
133     UINT64                  Address;            /* Physical addresss of the erro

141 } ACPI_TABLE_BERT;
```

unchanged_portion_omitted_

```
253 /* Masks for Flags field above */

255 #define ACPI_EINJ_PRESERVE          (1)

257 /* Values for Action field above */

259 enum AcpiEinjActions
260 {
261     ACPI_EINJ_BEGIN_OPERATION               = 0,
262     ACPI_EINJ_GET_TRIGGER_TABLE             = 1,
263     ACPI_EINJ_SET_ERROR_TYPE                = 2,
264     ACPI_EINJ_GET_ERROR_TYPE                = 3,
265     ACPI_EINJ_END_OPERATION                 = 4,
266     ACPI_EINJ_EXECUTE_OPERATION             = 5,
267     ACPI_EINJ_CHECK_BUSY_STATUS             = 6,
268     ACPI_EINJ_GET_COMMAND_STATUS            = 7,
269     ACPI_EINJ_SET_ERROR_TYPE_WITH_ADDRESS   = 8,
270     ACPI_EINJ_ACTION_RESERVED               = 9,     /* 9 and greater are reserv
263     ACPI_EINJ_ACTION_RESERVED       = 8,     /* 8 and greater are reserved */
271     ACPI_EINJ_TRIGGER_ERROR                 = 0xFF   /* Except for this value */
272 };

274 /* Values for Instruction field above */

276 enum AcpiEinjInstructions
277 {
278     ACPI_EINJ_READ_REGISTER         = 0,
279     ACPI_EINJ_READ_REGISTER_VALUE   = 1,
280     ACPI_EINJ_WRITE_REGISTER        = 2,
281     ACPI_EINJ_WRITE_REGISTER_VALUE  = 3,
282     ACPI_EINJ_NOOP                  = 4,
283     ACPI_EINJ_FLUSH_CACHELINE       = 5,
284     ACPI_EINJ_INSTRUCTION_RESERVED  = 6     /* 6 and greater are reserved */
276     ACPI_EINJ_INSTRUCTION_RESERVED  = 5     /* 5 and greater are reserved */
285 };

287 typedef struct acpi_einj_error_type_with_addr
288 {
289     UINT32                  ErrorType;
290     UINT32                  VendorStructOffset;
291     UINT32                  Flags;
292     UINT32                  ApicId;
293     UINT64                  Address;
294     UINT64                  Range;
295     UINT32                  PcieId;

297 } ACPI_EINJ_ERROR_TYPE_WITH_ADDR;

299 typedef struct acpi_einj_vendor
300 {
301     UINT32                  Length;
302     UINT32                  PcieId;
303     UINT16                  VendorId;
304     UINT16                  DeviceId;
305     UINT8                   RevisionId;
306     UINT8                   Reserved[3];
```



```
308 } ACPI_EINJ_VENDOR;


311 /* EINJ Trigger Error Action Table */

313 typedef struct acpi_einj_trigger
314 {
315     UINT32                  HeaderSize;
316     UINT32                  Revision;
317     UINT32                  TableSize;
318     UINT32                  EntryCount;

320 } ACPI_EINJ_TRIGGER;
```

unchanged_portion_omitted_

```
333 /* Error types returned from ACPI_EINJ_GET_ERROR_TYPE (bitfield) */

335 #define ACPI_EINJ_PROCESSOR_CORRECTABLE     (1)
336 #define ACPI_EINJ_PROCESSOR_UNCORRECTABLE   (1<<1)
337 #define ACPI_EINJ_PROCESSOR_FATAL           (1<<2)
338 #define ACPI_EINJ_MEMORY_CORRECTABLE        (1<<3)
339 #define ACPI_EINJ_MEMORY_UNCORRECTABLE      (1<<4)
340 #define ACPI_EINJ_MEMORY_FATAL              (1<<5)
341 #define ACPI_EINJ_PCIX_CORRECTABLE          (1<<6)
342 #define ACPI_EINJ_PCIX_UNCORRECTABLE        (1<<7)
343 #define ACPI_EINJ_PCIX_FATAL                (1<<8)
344 #define ACPI_EINJ_PLATFORM_CORRECTABLE      (1<<9)
345 #define ACPI_EINJ_PLATFORM_UNCORRECTABLE    (1<<10)
346 #define ACPI_EINJ_PLATFORM_FATAL            (1<<11)
347 #define ACPI_EINJ_VENDOR_DEFINED            (1<<31)


350 /******************************************************************************
351  *
352  * ERST - Error Record Serialization Table (ACPI 4.0)
353  *        Version 1
354  *
355  *****************************************************************************/

357 typedef struct acpi_table_erst
358 {
359     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
360     UINT32                  HeaderLength;
361     UINT32                  Reserved;
362     UINT32                  Entries;

364 } ACPI_TABLE_ERST;
```

unchanged_portion_omitted_

```
518 /* Common HEST sub-structure for PCI/AER structures below (6,7,8) */

520 typedef struct acpi_hest_aer_common
521 {
522     UINT16                  Reserved1;
523     UINT8                   Flags;
524     UINT8                   Enabled;
525     UINT32                  RecordsToPreallocate;
526     UINT32                  MaxSectionsPerRecord;
527     UINT32                  Bus;                    /* Bus and Segment numbers *
495     UINT32                  Bus;
528     UINT16                  Device;
529     UINT16                  Function;
530     UINT16                  DeviceControl;
531     UINT16                  Reserved2;
```

```
532     UINT32                  UncorrectableMask;
533     UINT32                  UncorrectableSeverity;
534     UINT32                  CorrectableMask;
535     UINT32                  AdvancedCapabilities;

537 } ACPI_HEST_AER_COMMON;

539 /* Masks for HEST Flags fields */

541 #define ACPI_HEST_FIRMWARE_FIRST        (1)
542 #define ACPI_HEST_GLOBAL                (1<<1)

544 /*
545  * Macros to access the bus/segment numbers in Bus field above:
546  *  Bus number is encoded in bits 7:0
547  *  Segment number is encoded in bits 23:8
548  */
549 #define ACPI_HEST_BUS(Bus)              ((Bus) & 0xFF)
550 #define ACPI_HEST_SEGMENT(Bus)          (((Bus) >> 8) & 0xFFFF)


553 /* Hardware Error Notification */

555 typedef struct acpi_hest_notify
556 {
557     UINT8                   Type;
558     UINT8                   Length;
559     UINT16                  ConfigWriteEnable;
560     UINT32                  PollInterval;
561     UINT32                  Vector;
562     UINT32                  PollingThresholdValue;
563     UINT32                  PollingThresholdWindow;
564     UINT32                  ErrorThresholdValue;
565     UINT32                  ErrorThresholdWindow;

567 } ACPI_HEST_NOTIFY;

569 /* Values for Notify Type field above */

571 enum AcpiHestNotifyTypes
572 {
573     ACPI_HEST_NOTIFY_POLLED     = 0,
574     ACPI_HEST_NOTIFY_EXTERNAL   = 1,
575     ACPI_HEST_NOTIFY_LOCAL      = 2,
576     ACPI_HEST_NOTIFY_SCI        = 3,
577     ACPI_HEST_NOTIFY_NMI        = 4,
578     ACPI_HEST_NOTIFY_CMCI       = 5,    /* ACPI 5.0 */
579     ACPI_HEST_NOTIFY_MCE        = 6,    /* ACPI 5.0 */
580     ACPI_HEST_NOTIFY_RESERVED   = 7     /* 7 and greater are reserved */
538     ACPI_HEST_NOTIFY_RESERVED   = 5     /* 5 and greater are reserved */
581 };
```

*_____unchanged_portion_omitted_*

```
751 /* Masks for Flags field above */

753 #define ACPI_MADT_PCAT_COMPAT       (1)         /* 00: System also has dual 8259

755 /* Values for PCATCompat flag */

757 #define ACPI_MADT_DUAL_PIC          0
758 #define ACPI_MADT_MULTIPLE_APIC     1


761 /* Values for MADT subtable type in ACPI_SUBTABLE_HEADER */

763 enum AcpiMadtType
```



```
764 {
765     ACPI_MADT_TYPE_LOCAL_APIC           = 0,
766     ACPI_MADT_TYPE_IO_APIC              = 1,
767     ACPI_MADT_TYPE_INTERRUPT_OVERRIDE   = 2,
768     ACPI_MADT_TYPE_NMI_SOURCE           = 3,
769     ACPI_MADT_TYPE_LOCAL_APIC_NMI       = 4,
770     ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE  = 5,
771     ACPI_MADT_TYPE_IO_SAPIC             = 6,
772     ACPI_MADT_TYPE_LOCAL_SAPIC          = 7,
773     ACPI_MADT_TYPE_INTERRUPT_SOURCE     = 8,
774     ACPI_MADT_TYPE_LOCAL_X2APIC         = 9,
775     ACPI_MADT_TYPE_LOCAL_X2APIC_NMI     = 10,
776     ACPI_MADT_TYPE_GENERIC_INTERRUPT    = 11,
777     ACPI_MADT_TYPE_GENERIC_DISTRIBUTOR  = 12,
778     ACPI_MADT_TYPE_RESERVED             = 13    /* 13 and greater are reserved *
734     ACPI_MADT_TYPE_RESERVED             = 11    /* 11 and greater are reserved *
779 };
```

*_____unchanged_portion_omitted_*

```
933 /* 11: Generic Interrupt (ACPI 5.0) */

935 typedef struct acpi_madt_generic_interrupt
936 {
937     ACPI_SUBTABLE_HEADER    Header;
938     UINT16                  Reserved;           /* Reserved - must be zero */
939     UINT32                  GicId;
940     UINT32                  Uid;
941     UINT32                  Flags;
942     UINT32                  ParkingVersion;
943     UINT32                  PerformanceInterrupt;
944     UINT64                  ParkedAddress;
945     UINT64                  BaseAddress;

947 } ACPI_MADT_GENERIC_INTERRUPT;


950 /* 12: Generic Distributor (ACPI 5.0) */

952 typedef struct acpi_madt_generic_distributor
953 {
954     ACPI_SUBTABLE_HEADER    Header;
955     UINT16                  Reserved;           /* Reserved - must be zero */
956     UINT32                  GicId;
957     UINT64                  BaseAddress;
958     UINT32                  GlobalIrqBase;
959     UINT32                  Reserved2;          /* Reserved - must be zero */

961 } ACPI_MADT_GENERIC_DISTRIBUTOR;


964 /*
965  * Common flags fields for MADT subtables
966  */

968 /* MADT Local APIC flags (LapicFlags) and GIC flags */
893 /* MADT Local APIC flags (LapicFlags) */

970 #define ACPI_MADT_ENABLED           (1)         /* 00: Processor is usable if se

972 /* MADT MPS INTI flags (IntiFlags) */

974 #define ACPI_MADT_POLARITY_MASK     (3)         /* 00-01: Polarity of APIC I/O i
975 #define ACPI_MADT_TRIGGER_MASK      (3<<2)      /* 02-03: Trigger mode of APIC i

977 /* Values for MPS INTI flags */
```

```
 979 #define ACPI_MADT_POLARITY_CONFORMS       0
 980 #define ACPI_MADT_POLARITY_ACTIVE_HIGH    1
 981 #define ACPI_MADT_POLARITY_RESERVED       2
 982 #define ACPI_MADT_POLARITY_ACTIVE_LOW     3

 984 #define ACPI_MADT_TRIGGER_CONFORMS        (0)
 985 #define ACPI_MADT_TRIGGER_EDGE            (1<<2)
 986 #define ACPI_MADT_TRIGGER_RESERVED        (2<<2)
 987 #define ACPI_MADT_TRIGGER_LEVEL           (3<<2)


 990 /*************************************************************************
 991  *
 992  * MSCT - Maximum System Characteristics Table (ACPI 4.0)
 993  *        Version 1
 994  *
 995  ************************************************************************/

 997 typedef struct acpi_table_msct
 998 {
 999     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
1000     UINT32                  ProximityOffset;    /* Location of proximity info st
1001     UINT32                  MaxProximityDomains;/* Max number of proximity domai
1002     UINT32                  MaxClockDomains;    /* Max number of clock domains *
1003     UINT64                  MaxAddress;         /* Max physical address in syste

1005 } ACPI_TABLE_MSCT;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   43681 Thu Dec 26 13:49:57 2013
new/usr/src/common/acpica/include/actbl2.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: actbl2.h - ACPI Table Definitions (tables not in ACPI spec)
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACTBL2_H__
  45 #define __ACTBL2_H__


  48 /******************************************************************************
  49  *
  50  * Additional ACPI Tables (2)
  51  *
  52  * These tables are not consumed directly by the ACPICA subsystem, but are
  53  * included here to support device drivers and the AML disassembler.
  54  *
  55  * The tables in this file are defined by third-party specifications, and are
  56  * not defined directly by the ACPI specification itself.
  57  *
  58  *****************************************************************************/
```



```
  61 /*
  62  * Values for description table header signatures for tables defined in this
  63  * file. Useful because they make it more difficult to inadvertently type in
  64  * the wrong signature.
  65  */
  66 #define ACPI_SIG_ASF            "ASF!"      /* Alert Standard Format table */
  67 #define ACPI_SIG_BOOT           "BOOT"      /* Simple Boot Flag Table */
  68 #define ACPI_SIG_CSRT           "CSRT"      /* Core System Resource Table */
  69 #define ACPI_SIG_DBG2           "DBG2"      /* Debug Port table type 2 */
  70 #define ACPI_SIG_DBGP           "DBGP"      /* Debug Port table */
  71 #define ACPI_SIG_DMAR           "DMAR"      /* DMA Remapping table */
  72 #define ACPI_SIG_HPET           "HPET"      /* High Precision Event Timer table
  73 #define ACPI_SIG_IBFT           "IBFT"      /* iSCSI Boot Firmware Table */
  74 #define ACPI_SIG_IVRS           "IVRS"      /* I/O Virtualization Reporting Stru
  75 #define ACPI_SIG_MCFG           "MCFG"      /* PCI Memory Mapped Configuration t
  76 #define ACPI_SIG_MCHI           "MCHI"      /* Management Controller Host Interf
  77 #define ACPI_SIG_MTMR           "MTMR"      /* MID Timer table */
  78 #define ACPI_SIG_SLIC           "SLIC"      /* Software Licensing Description Ta
  79 #define ACPI_SIG_SPCR           "SPCR"      /* Serial Port Console Redirection t
  80 #define ACPI_SIG_SPMI           "SPMI"      /* Server Platform Management Interf
  81 #define ACPI_SIG_TCPA           "TCPA"      /* Trusted Computing Platform Allian
  82 #define ACPI_SIG_UEFI           "UEFI"      /* Uefi Boot Optimization Table */
  83 #define ACPI_SIG_VRTC           "VRTC"      /* Virtual Real Time Clock Table */
  84 #define ACPI_SIG_WAET           "WAET"      /* Windows ACPI Emulated devices Tab
  85 #define ACPI_SIG_WDAT           "WDAT"      /* Watchdog Action Table */
  86 #define ACPI_SIG_WDDT           "WDDT"      /* Watchdog Timer Description Table
  87 #define ACPI_SIG_WDRT           "WDRT"      /* Watchdog Resource Table */

  89 #ifdef ACPI_UNDEFINED_TABLES
  90 /*
  91  * These tables have been seen in the field, but no definition has been found
  92  */
  93 #define ACPI_SIG_ATKG           "ATKG"
  94 #define ACPI_SIG_GSCI           "GSCI"      /* GMCH SCI table */
  95 #define ACPI_SIG_IEIT           "IEIT"
  96 #endif

  98 /*
  99  * All tables must be byte-packed to match the ACPI specification, since
 100  * the tables are provided by the system BIOS.
 101  */
 102 #pragma pack(1)

 104 /*
 105  * Note: C bitfields are not used for this reason:
 106  *
 107  * "Bitfields are great and easy to read, but unfortunately the C language
 108  * does not specify the layout of bitfields in memory, which means they are
 109  * essentially useless for dealing with packed data in on-disk formats or
 110  * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
 111  * this decision was a design error in C. Ritchie could have picked an order
 112  * and stuck with it." Norman Ramsey.
 113  * See http://stackoverflow.com/a/1053662/41661
 101  * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
 102  * This is the only type that is even remotely portable. Anything else is not
 103  * portable, so do not use any other bitfield types.
 114  */


 117 /******************************************************************************
 118  *
 119  * ASF - Alert Standard Format table (Signature "ASF!")
 120  *       Revision 0x10
 121  *
```

```
 122  * Conforms to the Alert Standard Format Specification V2.0, 23 April 2003
 123  *
 124  ******************************************************************************/

 126 typedef struct acpi_table_asf
 127 {
 128     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */

 130 } ACPI_TABLE_ASF;
_____unchanged_portion_omitted_


 275 /******************************************************************************
 276  *
 277  * CSRT - Core System Resource Table
 278  *        Version 0
 279  *
 280  * Conforms to the "Core System Resource Table (CSRT)", November 14, 2011
 281  *
 282  ******************************************************************************/

 284 typedef struct acpi_table_csrt
 285 {
 286     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */

 288 } ACPI_TABLE_CSRT;


 291 /* Resource Group subtable */

 293 typedef struct acpi_csrt_group
 294 {
 295     UINT32                  Length;
 296     UINT32                  VendorId;
 297     UINT32                  SubvendorId;
 298     UINT16                  DeviceId;
 299     UINT16                  SubdeviceId;
 300     UINT16                  Revision;
 301     UINT16                  Reserved;
 302     UINT32                  SharedInfoLength;

 304     /* Shared data immediately follows (Length = SharedInfoLength) */

 306 } ACPI_CSRT_GROUP;

 308 /* Shared Info subtable */

 310 typedef struct acpi_csrt_shared_info
 311 {
 312     UINT16                  MajorVersion;
 313     UINT16                  MinorVersion;
 314     UINT32                  MmioBaseLow;
 315     UINT32                  MmioBaseHigh;
 316     UINT32                  GsiInterrupt;
 317     UINT8                   InterruptPolarity;
 318     UINT8                   InterruptMode;
 319     UINT8                   NumChannels;
 320     UINT8                   DmaAddressWidth;
 321     UINT16                  BaseRequestLine;
 322     UINT16                  NumHandshakeSignals;
 323     UINT32                  MaxBlockSize;

 325     /* Resource descriptors immediately follow (Length = Group Length - SharedIn

 327 } ACPI_CSRT_SHARED_INFO;
```



```
 329 /* Resource Descriptor subtable */

 331 typedef struct acpi_csrt_descriptor
 332 {
 333     UINT32                  Length;
 334     UINT16                  Type;
 335     UINT16                  Subtype;
 336     UINT32                  Uid;

 338     /* Resource-specific information immediately follows */

 340 } ACPI_CSRT_DESCRIPTOR;


 343 /* Resource Types */

 345 #define ACPI_CSRT_TYPE_INTERRUPT    0x0001
 346 #define ACPI_CSRT_TYPE_TIMER        0x0002
 347 #define ACPI_CSRT_TYPE_DMA          0x0003

 349 /* Resource Subtypes */

 351 #define ACPI_CSRT_XRUPT_LINE        0x0000
 352 #define ACPI_CSRT_XRUPT_CONTROLLER  0x0001
 353 #define ACPI_CSRT_TIMER             0x0000
 354 #define ACPI_CSRT_DMA_CHANNEL       0x0000
 355 #define ACPI_CSRT_DMA_CONTROLLER    0x0001


 358 /******************************************************************************
 359  *
 360  * DBG2 - Debug Port Table 2
 361  *        Version 0 (Both main table and subtables)
 362  *
 363  * Conforms to "Microsoft Debug Port Table 2 (DBG2)", May 22 2012.
 364  *
 365  ******************************************************************************/

 367 typedef struct acpi_table_dbg2
 368 {
 369     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
 370     UINT32                  InfoOffset;
 371     UINT32                  InfoCount;

 373 } ACPI_TABLE_DBG2;


 376 typedef struct acpi_dbg2_header
 377 {
 378     UINT32                  InfoOffset;
 379     UINT32                  InfoCount;

 381 } ACPI_DBG2_HEADER;


 384 /* Debug Device Information Subtable */

 386 typedef struct acpi_dbg2_device
 387 {
 388     UINT8                   Revision;
 389     UINT16                  Length;
 390     UINT8                   RegisterCount;      /* Number of BaseAddress registe
 391     UINT16                  NamepathLength;
 392     UINT16                  NamepathOffset;
 393     UINT16                  OemDataLength;
 394     UINT16                  OemDataOffset;
```

```
395     UINT16                  PortType;
396     UINT16                  PortSubtype;
397     UINT16                  Reserved;
398     UINT16                  BaseAddressOffset;
399     UINT16                  AddressSizeOffset;
400     /*
401      * Data that follows:
402      *    BaseAddress (required) - Each in 12-byte Generic Address Structure for
403      *    AddressSize (required) - Array of UINT32 sizes corresponding to each B
404      *    Namepath    (required) - Null terminated string. Single dot if not sup
405      *    OemData     (optional) - Length is OemDataLength.
406      */
407 } ACPI_DBG2_DEVICE;

409 /* Types for PortType field above */

411 #define ACPI_DBG2_SERIAL_PORT       0x8000
412 #define ACPI_DBG2_1394_PORT         0x8001
413 #define ACPI_DBG2_USB_PORT          0x8002
414 #define ACPI_DBG2_NET_PORT          0x8003

416 /* Subtypes for PortSubtype field above */

418 #define ACPI_DBG2_16550_COMPATIBLE  0x0000
419 #define ACPI_DBG2_16550_SUBSET      0x0001

421 #define ACPI_DBG2_1394_STANDARD     0x0000

423 #define ACPI_DBG2_USB_XHCI          0x0000
424 #define ACPI_DBG2_USB_EHCI          0x0001


427 /*************************************************************************
428  *
429  * DBGP - Debug Port table
430  *        Version 1
431  *
432  * Conforms to the "Debug Port Specification", Version 1.00, 2/9/2000
433  *
434  ************************************************************************/

436 typedef struct acpi_table_dbgp
437 {
438     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
439     UINT8                   Type;               /* 0=full 16550, 1=subset of 165
440     UINT8                   Reserved[3];
441     ACPI_GENERIC_ADDRESS    DebugPort;

443 } ACPI_TABLE_DBGP;
_____unchanged_portion_omitted_


994 /*************************************************************************
995  *
996  * MTMR - MID Timer Table
997  *        Version 1
998  *
999  * Conforms to "Simple Firmware Interface Specification",
1000  * Draft 0.8.2, Oct 19, 2010
1001  * NOTE: The ACPI MTMR is equivalent to the SFI MTMR table.
1002  *
1003  ************************************************************************/

1005 typedef struct acpi_table_mtmr
1006 {
1007     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
```



```
1009 } ACPI_TABLE_MTMR;

1011 /* MTMR entry */

1013 typedef struct acpi_mtmr_entry
1014 {
1015     ACPI_GENERIC_ADDRESS    PhysicalAddress;
1016     UINT32                  Frequency;
1017     UINT32                  Irq;

1019 } ACPI_MTMR_ENTRY;


1022 /*************************************************************************
1023  *
1024  * SLIC - Software Licensing Description Table
1025  *        Version 1
1026  *
1027  * Conforms to "OEM Activation 2.0 for Windows Vista Operating Systems",
1028  * Copyright 2006
1029  *
1030  ************************************************************************/

1032 /* Basic SLIC table is only the common ACPI header */

1034 typedef struct acpi_table_slic
1035 {
1036     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */

1038 } ACPI_TABLE_SLIC;
_____unchanged_portion_omitted_


1221 /*************************************************************************
1222  *
1223  * VRTC - Virtual Real Time Clock Table
1224  *        Version 1
1225  *
1226  * Conforms to "Simple Firmware Interface Specification",
1227  * Draft 0.8.2, Oct 19, 2010
1228  * NOTE: The ACPI VRTC is equivalent to The SFI MRTC table.
1229  *
1230  ************************************************************************/

1232 typedef struct acpi_table_vrtc
1233 {
1234     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */

1236 } ACPI_TABLE_VRTC;

1238 /* VRTC entry */

1240 typedef struct acpi_vrtc_entry
1241 {
1242     ACPI_GENERIC_ADDRESS    PhysicalAddress;
1243     UINT32                  Irq;

1245 } ACPI_VRTC_ENTRY;


1248 /*************************************************************************
1249  *
1250  * WAET - Windows ACPI Emulated devices Table
1251  *        Version 1
1252  *
```

```
1253  * Conforms to "Windows ACPI Emulated Devices Table", version 1.0, April 6, 2009
1254  *
1255  ******************************************************************************/

1257 typedef struct acpi_table_waet
1258 {
1259     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
1260     UINT32                  Flags;

1262 } ACPI_TABLE_WAET;
```

_____***unchanged_portion_omitted_***

```
1419 /* Reset to default packing */

1421 #pragma pack()

1423 #endif /* __ACTBL2_H__ */
```

```
**********************************************************
   20829 Thu Dec 26 13:49:57 2013
new/usr/src/common/acpica/include/actbl3.h
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Name: actbl3.h - ACPI Table Definitions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACTBL3_H__
#define __ACTBL3_H__


/******************************************************************************
 *
 * Additional ACPI Tables (3)
 *
 * These tables are not consumed directly by the ACPICA subsystem, but are
 * included here to support device drivers and the AML disassembler.
 *
 * The tables in this file are fully defined within the ACPI specification.
 *
 *****************************************************************************/


/*
```



```
 * Values for description table header signatures for tables defined in this
 * file. Useful because they make it more difficult to inadvertently type in
 * the wrong signature.
 */
#define ACPI_SIG_BGRT           "BGRT"      /* Boot Graphics Resource Table */
#define ACPI_SIG_DRTM           "DRTM"      /* Dynamic Root of Trust for Measure
#define ACPI_SIG_FPDT           "FPDT"      /* Firmware Performance Data Table *
#define ACPI_SIG_GTDT           "GTDT"      /* Generic Timer Description Table *
#define ACPI_SIG_MPST           "MPST"      /* Memory Power State Table */
#define ACPI_SIG_PCCT           "PCCT"      /* Platform Communications Channel T
#define ACPI_SIG_PMTT           "PMTT"      /* Platform Memory Topology Table */
#define ACPI_SIG_RASF           "RASF"      /* RAS Feature table */
#define ACPI_SIG_TPM2           "TPM2"      /* Trusted Platform Module 2.0 H/W i

#define ACPI_SIG_S3PT           "S3PT"      /* S3 Performance (sub)Table */
#define ACPI_SIG_PCCS           "PCC"       /* PCC Shared Memory Region */

/* Reserved table signatures */

#define ACPI_SIG_MATR           "MATR"      /* Memory Address Translation Table
#define ACPI_SIG_MSDM           "MSDM"      /* Microsoft Data Management Table *
#define ACPI_SIG_WPBT           "WPBT"      /* Windows Platform Binary Table */

/*
 * All tables must be byte-packed to match the ACPI specification, since
 * the tables are provided by the system BIOS.
 */
#pragma pack(1)

/*
 * Note: C bitfields are not used for this reason:
 *
 * "Bitfields are great and easy to read, but unfortunately the C language
 * does not specify the layout of bitfields in memory, which means they are
 * essentially useless for dealing with packed data in on-disk formats or
 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
 * this decision was a design error in C. Ritchie could have picked an order
 * and stuck with it." Norman Ramsey.
 * See http://stackoverflow.com/a/1053662/41661
 */


/******************************************************************************
 *
 * BGRT - Boot Graphics Resource Table (ACPI 5.0)
 *        Version 1
 *
 *****************************************************************************/

typedef struct acpi_table_bgrt
{
    ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
    UINT16                  Version;
    UINT8                   Status;
    UINT8                   ImageType;
    UINT64                  ImageAddress;
    UINT32                  ImageOffsetX;
    UINT32                  ImageOffsetY;

} ACPI_TABLE_BGRT;


/******************************************************************************
 *
 * DRTM - Dynamic Root of Trust for Measurement table
 *
```

```
 *****************************************************************************/

typedef struct acpi_table_drtm
{
    ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
    UINT64                  EntryBaseAddress;
    UINT64                  EntryLength;
    UINT32                  EntryAddress32;
    UINT64                  EntryAddress64;
    UINT64                  ExitAddress;
    UINT64                  LogAreaAddress;
    UINT32                  LogAreaLength;
    UINT64                  ArchDependentAddress;
    UINT32                  Flags;

} ACPI_TABLE_DRTM;

/* 1) Validated Tables List */

typedef struct acpi_drtm_vtl_list
{
    UINT32                  ValidatedTableListCount;

} ACPI_DRTM_VTL_LIST;

/* 2) Resources List */

typedef struct acpi_drtm_resource_list
{
    UINT32                  ResourceListCount;

} ACPI_DRTM_RESOURCE_LIST;

/* 3) Platform-specific Identifiers List */

typedef struct acpi_drtm_id_list
{
    UINT32                  IdListCount;

} ACPI_DRTM_ID_LIST;


/*****************************************************************************
 *
 * FPDT - Firmware Performance Data Table (ACPI 5.0)
 *        Version 1
 *
 *****************************************************************************/

typedef struct acpi_table_fpdt
{
    ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */

} ACPI_TABLE_FPDT;


/* FPDT subtable header */

typedef struct acpi_fpdt_header
{
    UINT16                  Type;
    UINT8                   Length;
    UINT8                   Revision;

} ACPI_FPDT_HEADER;
```



```
/* Values for Type field above */

enum AcpiFpdtType
{
    ACPI_FPDT_TYPE_BOOT                 = 0,
    ACPI_FPDT_TYPE_S3PERF               = 1
};


/*
 * FPDT subtables
 */

/* 0: Firmware Basic Boot Performance Record */

typedef struct acpi_fpdt_boot
{
    ACPI_FPDT_HEADER        Header;
    UINT8                   Reserved[4];
    UINT64                  ResetEnd;
    UINT64                  LoadStart;
    UINT64                  StartupStart;
    UINT64                  ExitServicesEntry;
    UINT64                  ExitServicesExit;

} ACPI_FPDT_BOOT;


/* 1: S3 Performance Table Pointer Record */

typedef struct acpi_fpdt_s3pt_ptr
{
    ACPI_FPDT_HEADER        Header;
    UINT8                   Reserved[4];
    UINT64                  Address;

} ACPI_FPDT_S3PT_PTR;


/*
 * S3PT - S3 Performance Table. This table is pointed to by the
 * FPDT S3 Pointer Record above.
 */
typedef struct acpi_table_s3pt
{
    UINT8                   Signature[4]; /* "S3PT" */
    UINT32                  Length;

} ACPI_TABLE_S3PT;


/*
 * S3PT Subtables
 */
typedef struct acpi_s3pt_header
{
    UINT16                  Type;
    UINT8                   Length;
    UINT8                   Revision;

} ACPI_S3PT_HEADER;

/* Values for Type field above */

enum AcpiS3ptType
{
```

```
259     ACPI_S3PT_TYPE_RESUME               = 0,
260     ACPI_S3PT_TYPE_SUSPEND              = 1
261 };

263 typedef struct acpi_s3pt_resume
264 {
265     ACPI_S3PT_HEADER        Header;
266     UINT32                  ResumeCount;
267     UINT64                  FullResume;
268     UINT64                  AverageResume;

270 } ACPI_S3PT_RESUME;

272 typedef struct acpi_s3pt_suspend
273 {
274     ACPI_S3PT_HEADER        Header;
275     UINT64                  SuspendStart;
276     UINT64                  SuspendEnd;

278 } ACPI_S3PT_SUSPEND;


281 /*************************************************************************
282  *
283  * GTDT - Generic Timer Description Table (ACPI 5.0)
284  *        Version 1
285  *
286  ************************************************************************/

288 typedef struct acpi_table_gtdt
289 {
290     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
291     UINT64                  Address;
292     UINT32                  Flags;
293     UINT32                  SecurePl1Interrupt;
294     UINT32                  SecurePl1Flags;
295     UINT32                  NonSecurePl1Interrupt;
296     UINT32                  NonSecurePl1Flags;
297     UINT32                  VirtualTimerInterrupt;
298     UINT32                  VirtualTimerFlags;
299     UINT32                  NonSecurePl2Interrupt;
300     UINT32                  NonSecurePl2Flags;

302 } ACPI_TABLE_GTDT;

304 /* Values for Flags field above */

306 #define ACPI_GTDT_MAPPED_BLOCK_PRESENT      1

308 /* Values for all "TimerFlags" fields above */

310 #define ACPI_GTDT_INTERRUPT_MODE            1
311 #define ACPI_GTDT_INTERRUPT_POLARITY        2


314 /*************************************************************************
315  *
316  * MPST - Memory Power State Table (ACPI 5.0)
317  *        Version 1
318  *
319  ************************************************************************/

321 #define ACPI_MPST_CHANNEL_INFO \
322     UINT8                   ChannelId; \
323     UINT8                   Reserved1[3]; \
324     UINT16                  PowerNodeCount; \
```



```
325     UINT16                  Reserved2;

327 /* Main table */

329 typedef struct acpi_table_mpst
330 {
331     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
332     ACPI_MPST_CHANNEL_INFO                      /* Platform Communication Channe

334 } ACPI_TABLE_MPST;


337 /* Memory Platform Communication Channel Info */

339 typedef struct acpi_mpst_channel
340 {
341     ACPI_MPST_CHANNEL_INFO                      /* Platform Communication Channe

343 } ACPI_MPST_CHANNEL;


346 /* Memory Power Node Structure */

348 typedef struct acpi_mpst_power_node
349 {
350     UINT8                   Flags;
351     UINT8                   Reserved1;
352     UINT16                  NodeId;
353     UINT32                  Length;
354     UINT64                  RangeAddress;
355     UINT64                  RangeLength;
356     UINT32                  NumPowerStates;
357     UINT32                  NumPhysicalComponents;

359 } ACPI_MPST_POWER_NODE;

361 /* Values for Flags field above */

363 #define ACPI_MPST_ENABLED               1
364 #define ACPI_MPST_POWER_MANAGED         2
365 #define ACPI_MPST_HOT_PLUG_CAPABLE      4


368 /* Memory Power State Structure (follows POWER_NODE above) */

370 typedef struct acpi_mpst_power_state
371 {
372     UINT8                   PowerState;
373     UINT8                   InfoIndex;

375 } ACPI_MPST_POWER_STATE;


378 /* Physical Component ID Structure (follows POWER_STATE above) */

380 typedef struct acpi_mpst_component
381 {
382     UINT16                  ComponentId;

384 } ACPI_MPST_COMPONENT;


387 /* Memory Power State Characteristics Structure (follows all POWER_NODEs) */

389 typedef struct acpi_mpst_data_hdr
390 {
```

```
391     UINT16                  CharacteristicsCount;
392     UINT16                  Reserved;

394 } ACPI_MPST_DATA_HDR;

396 typedef struct acpi_mpst_power_data
397 {
398     UINT8                   StructureId;
399     UINT8                   Flags;
400     UINT16                  Reserved1;
401     UINT32                  AveragePower;
402     UINT32                  PowerSaving;
403     UINT64                  ExitLatency;
404     UINT64                  Reserved2;

406 } ACPI_MPST_POWER_DATA;

408 /* Values for Flags field above */

410 #define ACPI_MPST_PRESERVE              1
411 #define ACPI_MPST_AUTOENTRY             2
412 #define ACPI_MPST_AUTOEXIT              4


415 /* Shared Memory Region (not part of an ACPI table) */

417 typedef struct acpi_mpst_shared
418 {
419     UINT32                  Signature;
420     UINT16                  PccCommand;
421     UINT16                  PccStatus;
422     UINT32                  CommandRegister;
423     UINT32                  StatusRegister;
424     UINT32                  PowerStateId;
425     UINT32                  PowerNodeId;
426     UINT64                  EnergyConsumed;
427     UINT64                  AveragePower;

429 } ACPI_MPST_SHARED;


432 /*******************************************************************************
433  *
434  * PCCT - Platform Communications Channel Table (ACPI 5.0)
435  *        Version 1
436  *
437  ******************************************************************************/

439 typedef struct acpi_table_pcct
440 {
441     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
442     UINT32                  Flags;
443     UINT64                  Reserved;

445 } ACPI_TABLE_PCCT;

447 /* Values for Flags field above */

449 #define ACPI_PCCT_DOORBELL              1

451 /* Values for subtable type in ACPI_SUBTABLE_HEADER */

453 enum AcpiPcctType
454 {
455     ACPI_PCCT_TYPE_GENERIC_SUBSPACE     = 0,
456     ACPI_PCCT_TYPE_RESERVED             = 1     /* 1 and greater are reserved */
```



```
457 };

459 /*
460  * PCCT Subtables, correspond to Type in ACPI_SUBTABLE_HEADER
461  */

463 /* 0: Generic Communications Subspace */

465 typedef struct acpi_pcct_subspace
466 {
467     ACPI_SUBTABLE_HEADER    Header;
468     UINT8                   Reserved[6];
469     UINT64                  BaseAddress;
470     UINT64                  Length;
471     ACPI_GENERIC_ADDRESS    DoorbellRegister;
472     UINT64                  PreserveMask;
473     UINT64                  WriteMask;
474     UINT32                  Latency;
475     UINT32                  MaxAccessRate;
476     UINT16                  MinTurnaroundTime;

478 } ACPI_PCCT_SUBSPACE;


481 /*
482  * PCC memory structures (not part of the ACPI table)
483  */

485 /* Shared Memory Region */

487 typedef struct acpi_pcct_shared_memory
488 {
489     UINT32                  Signature;
490     UINT16                  Command;
491     UINT16                  Status;

493 } ACPI_PCCT_SHARED_MEMORY;


496 /*******************************************************************************
497  *
498  * PMTT - Platform Memory Topology Table (ACPI 5.0)
499  *        Version 1
500  *
501  ******************************************************************************/

503 typedef struct acpi_table_pmtt
504 {
505     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
506     UINT32                  Reserved;

508 } ACPI_TABLE_PMTT;


511 /* Common header for PMTT subtables that follow main table */

513 typedef struct acpi_pmtt_header
514 {
515     UINT8                   Type;
516     UINT8                   Reserved1;
517     UINT16                  Length;
518     UINT16                  Flags;
519     UINT16                  Reserved2;

521 } ACPI_PMTT_HEADER;
```

```
523 /* Values for Type field above */

525 #define ACPI_PMTT_TYPE_SOCKET           0
526 #define ACPI_PMTT_TYPE_CONTROLLER       1
527 #define ACPI_PMTT_TYPE_DIMM             2
528 #define ACPI_PMTT_TYPE_RESERVED         3 /* 0x03-0xFF are reserved */

530 /* Values for Flags field above */

532 #define ACPI_PMTT_TOP_LEVEL             0x0001
533 #define ACPI_PMTT_PHYSICAL              0x0002
534 #define ACPI_PMTT_MEMORY_TYPE           0x000C


537 /*
538  * PMTT subtables, correspond to Type in acpi_pmtt_header
539  */


542 /* 0: Socket Structure */

544 typedef struct acpi_pmtt_socket
545 {
546     ACPI_PMTT_HEADER        Header;
547     UINT16                  SocketId;
548     UINT16                  Reserved;

550 } ACPI_PMTT_SOCKET;


553 /* 1: Memory Controller subtable */

555 typedef struct acpi_pmtt_controller
556 {
557     ACPI_PMTT_HEADER        Header;
558     UINT32                  ReadLatency;
559     UINT32                  WriteLatency;
560     UINT32                  ReadBandwidth;
561     UINT32                  WriteBandwidth;
562     UINT16                  AccessWidth;
563     UINT16                  Alignment;
564     UINT16                  Reserved;
565     UINT16                  DomainCount;

567 } ACPI_PMTT_CONTROLLER;

569 /* 1a: Proximity Domain substructure */

571 typedef struct acpi_pmtt_domain
572 {
573     UINT32                  ProximityDomain;

575 } ACPI_PMTT_DOMAIN;


578 /* 2: Physical Component Identifier (DIMM) */

580 typedef struct acpi_pmtt_physical_component
581 {
582     ACPI_PMTT_HEADER        Header;
583     UINT16                  ComponentId;
584     UINT16                  Reserved;
585     UINT32                  MemorySize;
586     UINT32                  BiosHandle;

588 } ACPI_PMTT_PHYSICAL_COMPONENT;
```



```


591 /*******************************************************************************
592  *
593  * RASF - RAS Feature Table (ACPI 5.0)
594  *        Version 1
595  *
596  ******************************************************************************/

598 typedef struct acpi_table_rasf
599 {
600     ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
601     UINT8                   ChannelId[12];

603 } ACPI_TABLE_RASF;

605 /* RASF Platform Communication Channel Shared Memory Region */

607 typedef struct acpi_rasf_shared_memory
608 {
609     UINT32                  Signature;
610     UINT16                  Command;
611     UINT16                  Status;
612     UINT16                  Version;
613     UINT8                   Capabilities[16];
614     UINT8                   SetCapabilities[16];
615     UINT16                  NumParameterBlocks;
616     UINT32                  SetCapabilitiesStatus;

618 } ACPI_RASF_SHARED_MEMORY;

620 /* RASF Parameter Block Structure Header */

622 typedef struct acpi_rasf_parameter_block
623 {
624     UINT16                  Type;
625     UINT16                  Version;
626     UINT16                  Length;

628 } ACPI_RASF_PARAMETER_BLOCK;

630 /* RASF Parameter Block Structure for PATROL_SCRUB */

632 typedef struct acpi_rasf_patrol_scrub_parameter
633 {
634     ACPI_RASF_PARAMETER_BLOCK   Header;
635     UINT16                      PatrolScrubCommand;
636     UINT64                      RequestedAddressRange[2];
637     UINT64                      ActualAddressRange[2];
638     UINT16                      Flags;
639     UINT8                       RequestedSpeed;

641 } ACPI_RASF_PATROL_SCRUB_PARAMETER;

643 /* Masks for Flags and Speed fields above */

645 #define ACPI_RASF_SCRUBBER_RUNNING      1
646 #define ACPI_RASF_SPEED                 (7<<1)
647 #define ACPI_RASF_SPEED_SLOW            (0<<1)
648 #define ACPI_RASF_SPEED_MEDIUM          (4<<1)
649 #define ACPI_RASF_SPEED_FAST            (7<<1)

651 /* Channel Commands */

653 enum AcpiRasfCommands
654 {
```

```
    ACPI_RASF_EXECUTE_RASF_COMMAND      = 1
};

/* Platform RAS Capabilities */

enum AcpiRasfCapabiliities
{
    ACPI_HW_PATROL_SCRUB_SUPPORTED      = 0,
    ACPI_SW_PATROL_SCRUB_EXPOSED        = 1
};

/* Patrol Scrub Commands */

enum AcpiRasfPatrolScrubCommands
{
    ACPI_RASF_GET_PATROL_PARAMETERS     = 1,
    ACPI_RASF_START_PATROL_SCRUBBER     = 2,
    ACPI_RASF_STOP_PATROL_SCRUBBER      = 3
};

/* Channel Command flags */

#define ACPI_RASF_GENERATE_SCI          (1<<15)

/* Status values */

enum AcpiRasfStatus
{
    ACPI_RASF_SUCCESS                   = 0,
    ACPI_RASF_NOT_VALID                 = 1,
    ACPI_RASF_NOT_SUPPORTED             = 2,
    ACPI_RASF_BUSY                      = 3,
    ACPI_RASF_FAILED                    = 4,
    ACPI_RASF_ABORTED                   = 5,
    ACPI_RASF_INVALID_DATA              = 6
};

/* Status flags */

#define ACPI_RASF_COMMAND_COMPLETE      (1)
#define ACPI_RASF_SCI_DOORBELL          (1<<1)
#define ACPI_RASF_ERROR                 (1<<2)
#define ACPI_RASF_STATUS                (0x1F<<3)


/*******************************************************************************
 *
 * TPM2 - Trusted Platform Module (TPM) 2.0 Hardware Interface Table
 *        Version 3
 *
 * Conforms to "TPM 2.0 Hardware Interface Table (TPM2)" 29 November 2011
 *
 ******************************************************************************/

typedef struct acpi_table_tpm2
{
    ACPI_TABLE_HEADER       Header;             /* Common ACPI table header */
    UINT32                  Flags;
    UINT64                  ControlAddress;
    UINT32                  StartMethod;

} ACPI_TABLE_TPM2;

/* Control area structure (not part of table, pointed to by ControlAddress) */

typedef struct acpi_tpm2_control
```



```
{
    UINT32                  Reserved;
    UINT32                  Error;
    UINT32                  Cancel;
    UINT32                  Start;
    UINT64                  InterruptControl;
    UINT32                  CommandSize;
    UINT64                  CommandAddress;
    UINT32                  ResponseSize;
    UINT64                  ResponseAddress;

} ACPI_TPM2_CONTROL;


/* Reset to default packing */

#pragma pack()

#endif /* __ACTBL3_H__ */
```

```
**********************************************************
   45240 Thu Dec 26 13:49:57 2013
new/usr/src/common/acpica/include/actypes.h
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: actypes.h - Common data types for the entire ACPI subsystem
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACTYPES_H__
  45 #define __ACTYPES_H__

  47 /* acpisrc:StructDefs -- for acpisrc conversion */

  49 /*
  50  * ACPI_MACHINE_WIDTH must be specified in an OS- or compiler-dependent header
  51  * and must be either 32 or 64. 16-bit ACPICA is no longer supported, as of
  52  * 12/2006.
  53  */
  54 #ifndef ACPI_MACHINE_WIDTH
  55 #error ACPI_MACHINE_WIDTH not defined
  56 #endif
```



```
  58 /*! [Begin] no source code translation */

  60 /*
  61  * Data type ranges
  62  * Note: These macros are designed to be compiler independent as well as
  63  * working around problems that some 32-bit compilers have with 64-bit
  64  * constants.
  65  */
  66 #define ACPI_UINT8_MAX                  (UINT8) (~((UINT8)  0)) /* 0xFF
  67 #define ACPI_UINT16_MAX                 (UINT16)(~((UINT16) 0)) /* 0xFFFF
  68 #define ACPI_UINT32_MAX                 (UINT32)(~((UINT32) 0)) /* 0xFFFFFFFF
  69 #define ACPI_UINT64_MAX                 (UINT64)(~((UINT64) 0)) /* 0xFFFFFFFFFFFF
  70 #define ACPI_ASCII_MAX                  0x7F


  73 /*
  74  * Architecture-specific ACPICA Subsystem Data Types
  75  *
  76  * The goal of these types is to provide source code portability across
  77  * 16-bit, 32-bit, and 64-bit targets.
  78  *
  79  * 1) The following types are of fixed size for all targets (16/32/64):
  80  *
  81  * BOOLEAN      Logical boolean
  82  *
  83  * UINT8        8-bit  (1 byte) unsigned value
  84  * UINT16       16-bit (2 byte) unsigned value
  85  * UINT32       32-bit (4 byte) unsigned value
  86  * UINT64       64-bit (8 byte) unsigned value
  87  *
  88  * INT16        16-bit (2 byte) signed value
  89  * INT32        32-bit (4 byte) signed value
  90  * INT64        64-bit (8 byte) signed value
  91  *
  92  * COMPILER_DEPENDENT_UINT64/INT64 - These types are defined in the
  93  * compiler-dependent header(s) and were introduced because there is no common
  94  * 64-bit integer type across the various compilation models, as shown in
  95  * the table below.
  96  *
  97  * Datatype  LP64 ILP64 LLP64 ILP32 LP32 16bit
  98  * char      8    8     8     8     8    8
  99  * short     16   16    16    16    16   16
 100  * _int32          32
 101  * int       32   64    32    32    16   16
 102  * long      64   64    32    32    32   32
 103  * long long            64    64
 104  * pointer   64   64    64    32    32   32
 105  *
 106  * Note: ILP64 and LP32 are currently not supported.
 107  *
 108  *
 109  * 2) These types represent the native word size of the target mode of the
 110  * processor, and may be 16-bit, 32-bit, or 64-bit as required. They are
 111  * usually used for memory allocation, efficient loop counters, and array
 112  * indexes. The types are similar to the size_t type in the C library and are
 113  * required because there is no C type that consistently represents the native
 114  * data width. ACPI_SIZE is needed because there is no guarantee that a
 115  * kernel-level C library is present.
 116  *
 117  * ACPI_SIZE        16/32/64-bit unsigned value
 118  * ACPI_NATIVE_INT  16/32/64-bit signed value
 119  */

 121 /******************************************************************************
 122  *
 123  * Common types for all compilers, all targets
```

```
 *
 *****************************************************************************/

typedef unsigned char                   BOOLEAN;
typedef unsigned char                   UINT8;
typedef unsigned short                  UINT16;
typedef COMPILER_DEPENDENT_UINT64       UINT64;
typedef COMPILER_DEPENDENT_INT64        INT64;

/*! [End] no source code translation !*/

/*
 * Value returned by AcpiOsGetThreadId. There is no standard "thread_id"
 * across operating systems or even the various UNIX systems. Since ACPICA
 * only needs the thread ID as a unique thread identifier, we use a UINT64
 * as the only common data type - it will accommodate any type of pointer or
 * any type of integer. It is up to the host-dependent OSL to cast the
 * native thread ID type to a UINT64 (in AcpiOsGetThreadId).
 */
#define ACPI_THREAD_ID                  UINT64


/*****************************************************************************
 *
 * Types specific to 64-bit targets
 *
 *****************************************************************************/

#if ACPI_MACHINE_WIDTH == 64

/*! [Begin] no source code translation (keep the typedefs as-is) */

typedef unsigned int                    UINT32;
typedef int                             INT32;

/*! [End] no source code translation !*/


typedef INT64                           ACPI_NATIVE_INT;
typedef UINT64                          ACPI_SIZE;
typedef UINT64                          ACPI_IO_ADDRESS;
typedef UINT64                          ACPI_PHYSICAL_ADDRESS;

#define ACPI_MAX_PTR                    ACPI_UINT64_MAX
#define ACPI_SIZE_MAX                   ACPI_UINT64_MAX
#define ACPI_USE_NATIVE_DIVIDE          /* Has native 64-bit integer support */

/*
 * In the case of the Itanium Processor Family (IPF), the hardware does not
 * support misaligned memory transfers. Set the MISALIGNMENT_NOT_SUPPORTED flag
 * to indicate that special precautions must be taken to avoid alignment faults.
 * (IA64 or ia64 is currently used by existing compilers to indicate IPF.)
 *
 * Note: EM64T and other X86-64 processors support misaligned transfers,
 * so there is no need to define this flag.
 */
#if defined (__IA64__) || defined (__ia64__)
#define ACPI_MISALIGNMENT_NOT_SUPPORTED
#endif


/*****************************************************************************
 *
 * Types specific to 32-bit targets
 *
 *****************************************************************************/
```



```
#elif ACPI_MACHINE_WIDTH == 32

/*! [Begin] no source code translation (keep the typedefs as-is) */

typedef unsigned int                    UINT32;
typedef int                             INT32;

/*! [End] no source code translation !*/


typedef INT32                           ACPI_NATIVE_INT;
typedef UINT32                          ACPI_SIZE;
typedef UINT32                          ACPI_IO_ADDRESS;
typedef UINT32                          ACPI_PHYSICAL_ADDRESS;

#define ACPI_MAX_PTR                    ACPI_UINT32_MAX
#define ACPI_SIZE_MAX                   ACPI_UINT32_MAX

#else

/* ACPI_MACHINE_WIDTH must be either 64 or 32 */

#error unknown ACPI_MACHINE_WIDTH
#endif


/*****************************************************************************
 *
 * OS-dependent types
 *
 * If the defaults below are not appropriate for the host system, they can
 * be defined in the OS-specific header, and this will take precedence.
 *
 *****************************************************************************/

/* Flags for AcpiOsAcquireLock/AcpiOsReleaseLock */

#ifndef ACPI_CPU_FLAGS
#define ACPI_CPU_FLAGS                  ACPI_SIZE
#endif

/* Object returned from AcpiOsCreateCache */

#ifndef ACPI_CACHE_T
#ifdef ACPI_USE_LOCAL_CACHE
#define ACPI_CACHE_T                    ACPI_MEMORY_LIST
#else
#define ACPI_CACHE_T                    void *
#endif
#endif

/*
 * Synchronization objects - Mutexes, Semaphores, and SpinLocks
 */
#if (ACPI_MUTEX_TYPE == ACPI_BINARY_SEMAPHORE)
/*
 * These macros are used if the host OS does not support a mutex object.
 * Map the OSL Mutex interfaces to binary semaphores.
 */
#define ACPI_MUTEX                      ACPI_SEMAPHORE
#define AcpiOsCreateMutex(OutHandle)    AcpiOsCreateSemaphore (1, 1, OutHandle)
#define AcpiOsDeleteMutex(Handle)       (void) AcpiOsDeleteSemaphore (Handle)
#define AcpiOsAcquireMutex(Handle,Time) AcpiOsWaitSemaphore (Handle, 1, Time)
#define AcpiOsReleaseMutex(Handle)      (void) AcpiOsSignalSemaphore (Handle, 1)
#endif
```

```
257 /* Configurable types for synchronization objects */

259 #ifndef ACPI_SPINLOCK
260 #define ACPI_SPINLOCK                   void *
261 #endif

263 #ifndef ACPI_SEMAPHORE
264 #define ACPI_SEMAPHORE                  void *
265 #endif

267 #ifndef ACPI_MUTEX
268 #define ACPI_MUTEX                      void *
269 #endif


272 /*****************************************************************************
273  *
274  * Compiler-dependent types
275  *
276  * If the defaults below are not appropriate for the host compiler, they can
277  * be defined in the compiler-specific header, and this will take precedence.
278  *
279  ****************************************************************************/

281 /* Use C99 uintptr_t for pointer casting if available, "void *" otherwise */

283 #ifndef ACPI_UINTPTR_T
284 #define ACPI_UINTPTR_T                  void *
285 #endif

287 /*
288  * ACPI_PRINTF_LIKE is used to tag functions as "printf-like" because
289  * some compilers can catch printf format string problems
290  */
291 #ifndef ACPI_PRINTF_LIKE
292 #define ACPI_PRINTF_LIKE(c)
293 #endif

295 /*
296  * Some compilers complain about unused variables. Sometimes we don't want to
297  * use all the variables (for example, _AcpiModuleName). This allows us
298  * to tell the compiler in a per-variable manner that a variable
298  * to tell the compiler in a per-variable manner that a variable
299  * is unused
300  */
301 #ifndef ACPI_UNUSED_VAR
302 #define ACPI_UNUSED_VAR
303 #endif

305 /*
306  * All ACPICA external functions that are available to the rest of the kernel
307  * are tagged with thes macros which can be defined as appropriate for the host.
308  *
309  * Notes:
310  * ACPI_EXPORT_SYMBOL_INIT is used for initialization and termination
311  * interfaces that may need special processing.
312  * ACPI_EXPORT_SYMBOL is used for all other public external functions.
306  * All ACPICA functions that are available to the rest of the kernel are
307  * tagged with this macro which can be defined as appropriate for the host.
313  */
314 #ifndef ACPI_EXPORT_SYMBOL_INIT
315 #define ACPI_EXPORT_SYMBOL_INIT(Symbol)
316 #endif

318 #ifndef ACPI_EXPORT_SYMBOL
```



```
319 #define ACPI_EXPORT_SYMBOL(Symbol)
320 #endif

322 /*
323  * Compiler/Clibrary-dependent debug initialization. Used for ACPICA
324  * utilities only.
325  */
326 #ifndef ACPI_DEBUG_INITIALIZE
327 #define ACPI_DEBUG_INITIALIZE()
328 #endif


331 /*****************************************************************************
332  *
333  * Configuration
334  *
335  ****************************************************************************/

337 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
338 /*
339  * Memory allocation tracking (used by AcpiExec to detect memory leaks)
340  */
341 #define ACPI_MEM_PARAMETERS             _COMPONENT, _AcpiModuleName, __LINE__
342 #define ACPI_ALLOCATE(a)                AcpiUtAllocateAndTrack ((ACPI_SIZE) (a),
343 #define ACPI_ALLOCATE_ZEROED(a)         AcpiUtAllocateZeroedAndTrack ((ACPI_SIZE
344 #define ACPI_FREE(a)                    AcpiUtFreeAndTrack (a, ACPI_MEM_PARAMETE
345 #define ACPI_MEM_TRACKING(a)            a

347 #else
348 /*
349  * Normal memory allocation directly via the OS services layer
350  */
351 #define ACPI_ALLOCATE(a)                AcpiOsAllocate ((ACPI_SIZE) (a))
352 #define ACPI_ALLOCATE_ZEROED(a)         AcpiOsAllocateZeroed ((ACPI_SIZE) (a))
353 #define ACPI_FREE(a)                    AcpiOsFree (a)
354 #define ACPI_MEM_TRACKING(a)

356 #endif /* ACPI_DBG_TRACK_ALLOCATIONS */


359 /*****************************************************************************
360  *
361  * ACPI Specification constants (Do not change unless the specification changes)
362  *
363  ****************************************************************************/

365 /* Number of distinct FADT-based GPE register blocks (GPE0 and GPE1) */

367 #define ACPI_MAX_GPE_BLOCKS             2

369 /* Default ACPI register widths */

371 #define ACPI_GPE_REGISTER_WIDTH         8
372 #define ACPI_PM1_REGISTER_WIDTH         16
373 #define ACPI_PM2_REGISTER_WIDTH         8
374 #define ACPI_PM_TIMER_WIDTH             32
375 #define ACPI_RESET_REGISTER_WIDTH       8

377 /* Names within the namespace are 4 bytes long */

379 #define ACPI_NAME_SIZE                  4
380 #define ACPI_PATH_SEGMENT_LENGTH        5           /* 4 chars for name + 1 char
381 #define ACPI_PATH_SEPARATOR             '.'

383 /* Sizes for ACPI table headers */
```

```
385 #define ACPI_OEM_ID_SIZE                6
386 #define ACPI_OEM_TABLE_ID_SIZE          8

388 /* ACPI/PNP hardware IDs */

390 #define PCI_ROOT_HID_STRING             "PNP0A03"
391 #define PCI_EXPRESS_ROOT_HID_STRING     "PNP0A08"

393 /* PM Timer ticks per second (HZ) */

395 #define ACPI_PM_TIMER_FREQUENCY         3579545
349 #define PM_TIMER_FREQUENCY              3579545


398 /*****************************************************************************
399  *
400  * Independent types
401  *
402  ****************************************************************************/

404 /* Logical defines and NULL */

406 #ifdef FALSE
407 #undef FALSE
408 #endif
409 #define FALSE                           (1 == 0)

411 #ifdef TRUE
412 #undef TRUE
413 #endif
414 #define TRUE                            (1 == 1)

416 #ifndef NULL
417 #define NULL                            (void *) 0
418 #endif


421 /*
422  * Miscellaneous types
423  */
424 typedef UINT32                          ACPI_STATUS;    /* All ACPI Exceptions *
425 typedef UINT32                          ACPI_NAME;      /* 4-byte ACPI name */
426 typedef char *                          ACPI_STRING;    /* Null terminated ASCII
427 typedef void *                          ACPI_HANDLE;    /* Actually a ptr to a N


430 /* Time constants for timer calculations */

432 #define ACPI_MSEC_PER_SEC               1000L

434 #define ACPI_USEC_PER_MSEC              1000L
435 #define ACPI_USEC_PER_SEC               1000000L

437 #define ACPI_100NSEC_PER_USEC           10L
438 #define ACPI_100NSEC_PER_MSEC           10000L
439 #define ACPI_100NSEC_PER_SEC            10000000L

441 #define ACPI_NSEC_PER_USEC              1000L
442 #define ACPI_NSEC_PER_MSEC              1000000L
443 #define ACPI_NSEC_PER_SEC               1000000000L


446 /* Owner IDs are used to track namespace nodes for selective deletion */

448 typedef UINT8                           ACPI_OWNER_ID;
449 #define ACPI_OWNER_ID_MAX               0xFF
```



```
452 #define ACPI_INTEGER_BIT_SIZE           64
453 #define ACPI_MAX_DECIMAL_DIGITS         20  /* 2^64 = 18,446,744,073,709,551,616
454 #define ACPI_MAX64_DECIMAL_DIGITS       20
455 #define ACPI_MAX32_DECIMAL_DIGITS       10
456 #define ACPI_MAX16_DECIMAL_DIGITS        5
457 #define ACPI_MAX8_DECIMAL_DIGITS         3

459 /*
460  * Constants with special meanings
461  */
462 #define ACPI_ROOT_OBJECT                ACPI_ADD_PTR (ACPI_HANDLE, NULL, ACPI_MA
463 #define ACPI_WAIT_FOREVER               0xFFFF  /* UINT16, as per ACPI spec */
464 #define ACPI_DO_NOT_WAIT                0

466 /*
467  * Obsolete: Acpi integer width. In ACPI version 1 (1996), integers are 32 bits.
468  * In ACPI version 2 (2000) and later, integers are 64 bits. Note that this
469  * pertains to the ACPI integer type only, not to other integers used in the
470  * implementation of the ACPICA subsystem.
471  *
472  * 01/2010: This type is obsolete and has been removed from the entire ACPICA
473  * code base. It remains here for compatibility with device drivers that use
474  * the type. However, it will be removed in the future.
475  */
476 typedef UINT64                          ACPI_INTEGER;
477 #define ACPI_INTEGER_MAX                ACPI_UINT64_MAX


480 /*****************************************************************************
481  *
482  * Commonly used macros
483  *
484  ****************************************************************************/

486 /* Data manipulation */

488 #define ACPI_LOBYTE(Integer)            ((UINT8)   (UINT16)(Integer))
489 #define ACPI_HIBYTE(Integer)            ((UINT8) (((UINT16)(Integer)) >> 8))
490 #define ACPI_LOWORD(Integer)            ((UINT16)  (UINT32)(Integer))
491 #define ACPI_HIWORD(Integer)            ((UINT16)(((UINT32)(Integer)) >> 16))
492 #define ACPI_LODWORD(Integer64)         ((UINT32)  (UINT64)(Integer64))
493 #define ACPI_HIDWORD(Integer64)         ((UINT32)(((UINT64)(Integer64)) >> 32))

495 #define ACPI_SET_BIT(target,bit)        ((target) |= (bit))
496 #define ACPI_CLEAR_BIT(target,bit)      ((target) &= ~(bit))
497 #define ACPI_MIN(a,b)                   (((a)<(b))?(a):(b))
498 #define ACPI_MAX(a,b)                   (((a)>(b))?(a):(b))

500 /* Size calculation */

502 #define ACPI_ARRAY_LENGTH(x)            (sizeof(x) / sizeof((x)[0]))

504 /* Pointer manipulation */

506 #define ACPI_CAST_PTR(t, p)             ((t *) (ACPI_UINTPTR_T) (p))
507 #define ACPI_CAST_INDIRECT_PTR(t, p)    ((t **) (ACPI_UINTPTR_T) (p))
508 #define ACPI_ADD_PTR(t, a, b)           ACPI_CAST_PTR (t, (ACPI_CAST_PTR (UINT8,
509 #define ACPI_PTR_DIFF(a, b)             (ACPI_SIZE) (ACPI_CAST_PTR (UINT8, (a))

511 /* Pointer/Integer type conversions */

513 #define ACPI_TO_POINTER(i)              ACPI_ADD_PTR (void, (void *) NULL,(ACPI_
514 #define ACPI_TO_INTEGER(p)              ACPI_PTR_DIFF (p, (void *) NULL)
515 #define ACPI_OFFSET(d, f)               (ACPI_SIZE) ACPI_PTR_DIFF (&(((d *)0)->f
```

```
516 #define ACPI_PHYSADDR_TO_PTR(i)         ACPI_TO_POINTER(i)
517 #define ACPI_PTR_TO_PHYSADDR(i)         ACPI_TO_INTEGER(i)

519 /* Optimizations for 4-character (32-bit) ACPI_NAME manipulation */

521 #ifndef ACPI_MISALIGNMENT_NOT_SUPPORTED
522 #define ACPI_COMPARE_NAME(a,b)          (*ACPI_CAST_PTR (UINT32, (a)) == *ACPI_C
523 #define ACPI_MOVE_NAME(dest,src)        (*ACPI_CAST_PTR (UINT32, (dest)) = *ACPI
524 #else
525 #define ACPI_COMPARE_NAME(a,b)          (!ACPI_STRNCMP (ACPI_CAST_PTR (char, (a)
526 #define ACPI_MOVE_NAME(dest,src)        (ACPI_STRNCPY (ACPI_CAST_PTR (char, (des
527 #endif

529 /* Support for the special RSDP signature (8 characters) */

531 #define ACPI_VALIDATE_RSDP_SIG(a)       (!ACPI_STRNCMP (ACPI_CAST_PTR (char, (a)
532 #define ACPI_MAKE_RSDP_SIG(dest)        (ACPI_MEMCPY (ACPI_CAST_PTR (char, (dest


535 /*******************************************************************************
536  *
537  * Miscellaneous constants
538  *
539  ******************************************************************************/

541 /*
542  * Initialization sequence
543  */
544 #define ACPI_FULL_INITIALIZATION        0x00
545 #define ACPI_NO_ADDRESS_SPACE_INIT      0x01
546 #define ACPI_NO_HARDWARE_INIT           0x02
547 #define ACPI_NO_EVENT_INIT              0x04
548 #define ACPI_NO_HANDLER_INIT            0x08
549 #define ACPI_NO_ACPI_ENABLE             0x10
550 #define ACPI_NO_DEVICE_INIT             0x20
551 #define ACPI_NO_OBJECT_INIT             0x40

553 /*
554  * Initialization state
555  */
556 #define ACPI_SUBSYSTEM_INITIALIZE       0x01
557 #define ACPI_INITIALIZED_OK             0x02

559 /*
560  * Power state values
561  */
562 #define ACPI_STATE_UNKNOWN              (UINT8) 0xFF

564 #define ACPI_STATE_S0                   (UINT8) 0
565 #define ACPI_STATE_S1                   (UINT8) 1
566 #define ACPI_STATE_S2                   (UINT8) 2
567 #define ACPI_STATE_S3                   (UINT8) 3
568 #define ACPI_STATE_S4                   (UINT8) 4
569 #define ACPI_STATE_S5                   (UINT8) 5
570 #define ACPI_S_STATES_MAX               ACPI_STATE_S5
571 #define ACPI_S_STATE_COUNT              6

573 #define ACPI_STATE_D0                   (UINT8) 0
574 #define ACPI_STATE_D1                   (UINT8) 1
575 #define ACPI_STATE_D2                   (UINT8) 2
576 #define ACPI_STATE_D3                   (UINT8) 3
577 #define ACPI_D_STATES_MAX               ACPI_STATE_D3
578 #define ACPI_D_STATE_COUNT              4

580 #define ACPI_STATE_C0                   (UINT8) 0
581 #define ACPI_STATE_C1                   (UINT8) 1
```



```
582 #define ACPI_STATE_C2                   (UINT8) 2
583 #define ACPI_STATE_C3                   (UINT8) 3
584 #define ACPI_C_STATES_MAX               ACPI_STATE_C3
585 #define ACPI_C_STATE_COUNT              4

587 /*
588  * Sleep type invalid value
589  */
590 #define ACPI_SLEEP_TYPE_MAX             0x7
591 #define ACPI_SLEEP_TYPE_INVALID         0xFF

593 /*
594  * Standard notify values
595  */
596 #define ACPI_NOTIFY_BUS_CHECK           (UINT8) 0x00
597 #define ACPI_NOTIFY_DEVICE_CHECK        (UINT8) 0x01
598 #define ACPI_NOTIFY_DEVICE_WAKE         (UINT8) 0x02
599 #define ACPI_NOTIFY_EJECT_REQUEST       (UINT8) 0x03
600 #define ACPI_NOTIFY_DEVICE_CHECK_LIGHT  (UINT8) 0x04
601 #define ACPI_NOTIFY_FREQUENCY_MISMATCH  (UINT8) 0x05
602 #define ACPI_NOTIFY_BUS_MODE_MISMATCH   (UINT8) 0x06
603 #define ACPI_NOTIFY_POWER_FAULT         (UINT8) 0x07
604 #define ACPI_NOTIFY_CAPABILITIES_CHECK  (UINT8) 0x08
605 #define ACPI_NOTIFY_DEVICE_PLD_CHECK    (UINT8) 0x09
606 #define ACPI_NOTIFY_RESERVED            (UINT8) 0x0A
607 #define ACPI_NOTIFY_LOCALITY_UPDATE     (UINT8) 0x0B
608 #define ACPI_NOTIFY_SHUTDOWN_REQUEST    (UINT8) 0x0C

610 #define ACPI_NOTIFY_MAX                 0x0C
538 #define ACPI_NOTIFY_MAX                 0x0B

612 /*
613  * Types associated with ACPI names and objects. The first group of
614  * values (up to ACPI_TYPE_EXTERNAL_MAX) correspond to the definition
615  * of the ACPI ObjectType() operator (See the ACPI Spec). Therefore,
616  * only add to the first group if the spec changes.
617  *
618  * NOTE: Types must be kept in sync with the global AcpiNsProperties
619  * and AcpiNsTypeNames arrays.
620  */
621 typedef UINT32                          ACPI_OBJECT_TYPE;

623 #define ACPI_TYPE_ANY                   0x00
624 #define ACPI_TYPE_INTEGER               0x01  /* Byte/Word/Dword/Zero/One/Ones *
625 #define ACPI_TYPE_STRING                0x02
626 #define ACPI_TYPE_BUFFER                0x03
627 #define ACPI_TYPE_PACKAGE               0x04  /* ByteConst, multiple DataTerm/Co
628 #define ACPI_TYPE_FIELD_UNIT            0x05
629 #define ACPI_TYPE_DEVICE                0x06  /* Name, multiple Node */
630 #define ACPI_TYPE_EVENT                 0x07
631 #define ACPI_TYPE_METHOD                0x08  /* Name, ByteConst, multiple Code
632 #define ACPI_TYPE_MUTEX                 0x09
633 #define ACPI_TYPE_REGION                0x0A
634 #define ACPI_TYPE_POWER                 0x0B  /* Name,ByteConst,WordConst,multi
635 #define ACPI_TYPE_PROCESSOR             0x0C  /* Name,ByteConst,DWordConst,ByteC
636 #define ACPI_TYPE_THERMAL               0x0D  /* Name, multiple Node */
637 #define ACPI_TYPE_BUFFER_FIELD          0x0E
638 #define ACPI_TYPE_DDB_HANDLE            0x0F
639 #define ACPI_TYPE_DEBUG_OBJECT          0x10

641 #define ACPI_TYPE_EXTERNAL_MAX          0x10

643 /*
644  * These are object types that do not map directly to the ACPI
645  * ObjectType() operator. They are used for various internal purposes only.
646  * If new predefined ACPI_TYPEs are added (via the ACPI specification), these
```

```
 647  * internal types must move upwards. (There is code that depends on these
 648  * values being contiguous with the external types above.)
 649  */
 650 #define ACPI_TYPE_LOCAL_REGION_FIELD    0x11
 651 #define ACPI_TYPE_LOCAL_BANK_FIELD      0x12
 652 #define ACPI_TYPE_LOCAL_INDEX_FIELD     0x13
 653 #define ACPI_TYPE_LOCAL_REFERENCE       0x14  /* Arg#, Local#, Name, Debug, RefO
 654 #define ACPI_TYPE_LOCAL_ALIAS           0x15
 655 #define ACPI_TYPE_LOCAL_METHOD_ALIAS    0x16
 656 #define ACPI_TYPE_LOCAL_NOTIFY          0x17
 657 #define ACPI_TYPE_LOCAL_ADDRESS_HANDLER 0x18
 658 #define ACPI_TYPE_LOCAL_RESOURCE        0x19
 659 #define ACPI_TYPE_LOCAL_RESOURCE_FIELD  0x1A
 660 #define ACPI_TYPE_LOCAL_SCOPE           0x1B  /* 1 Name, multiple ObjectList Nod

 662 #define ACPI_TYPE_NS_NODE_MAX           0x1B  /* Last typecode used within a NS

 664 /*
 665  * These are special object types that never appear in
 666  * a Namespace node, only in an object of ACPI_OPERAND_OBJECT
 594  * a Namespace node, only in an ACPI_OPERAND_OBJECT
 667  */
 668 #define ACPI_TYPE_LOCAL_EXTRA           0x1C
 669 #define ACPI_TYPE_LOCAL_DATA            0x1D

 671 #define ACPI_TYPE_LOCAL_MAX             0x1D

 673 /* All types above here are invalid */

 675 #define ACPI_TYPE_INVALID               0x1E
 676 #define ACPI_TYPE_NOT_FOUND             0xFF

 678 #define ACPI_NUM_NS_TYPES               (ACPI_TYPE_INVALID + 1)


 681 /*
 682  * All I/O
 683  */
 684 #define ACPI_READ                       0
 685 #define ACPI_WRITE                      1
 686 #define ACPI_IO_MASK                    1

 688 /*
 689  * Event Types: Fixed & General Purpose
 690  */
 691 typedef UINT32                          ACPI_EVENT_TYPE;

 693 /*
 694  * Fixed events
 695  */
 696 #define ACPI_EVENT_PMTIMER              0
 697 #define ACPI_EVENT_GLOBAL               1
 698 #define ACPI_EVENT_POWER_BUTTON         2
 699 #define ACPI_EVENT_SLEEP_BUTTON         3
 700 #define ACPI_EVENT_RTC                  4
 701 #define ACPI_EVENT_MAX                  4
 702 #define ACPI_NUM_FIXED_EVENTS           ACPI_EVENT_MAX + 1

 704 /*
 705  * Event Status - Per event
 706  * -------------
 707  * The encoding of ACPI_EVENT_STATUS is illustrated below.
 708  * Note that a set bit (1) indicates the property is TRUE
 709  * (e.g. if bit 0 is set then the event is enabled).
 710  * +-------------+-+-+-+
 711  * |   Bits 31:3 |2|1|0|
```



```
 712  * +-------------+-+-+-+
 713  *          |     | | |
 714  *          |     | | +- Enabled?
 715  *          |     | +--- Enabled for wake?
 716  *          |     +----- Set?
 717  *          +----------- <Reserved>
 718  */
 719 typedef UINT32                          ACPI_EVENT_STATUS;

 721 #define ACPI_EVENT_FLAG_DISABLED        (ACPI_EVENT_STATUS) 0x00
 722 #define ACPI_EVENT_FLAG_ENABLED         (ACPI_EVENT_STATUS) 0x01
 723 #define ACPI_EVENT_FLAG_WAKE_ENABLED    (ACPI_EVENT_STATUS) 0x02
 724 #define ACPI_EVENT_FLAG_SET             (ACPI_EVENT_STATUS) 0x04

 654 /*
 655  * General Purpose Events (GPE)
 656  */
 657 #define ACPI_GPE_INVALID                0xFF
 658 #define ACPI_GPE_MAX                    0xFF
 659 #define ACPI_NUM_GPE                    256

 726 /* Actions for AcpiSetGpe, AcpiGpeWakeup, AcpiHwLowSetGpe */

 728 #define ACPI_GPE_ENABLE                 0
 729 #define ACPI_GPE_DISABLE                1
 730 #define ACPI_GPE_CONDITIONAL_ENABLE     2

 732 /*
 733  * GPE info flags - Per GPE
 734  * +-------+-+-+---+
 735  * |  7:4  |3|2|1:0|
 736  * +-------+-+-+---+
 737  *     |    | |  |
 738  *     |    | |  +-- Type of dispatch:to method, handler, notify, or none
 739  *     |    | +----- Interrupt type: edge or level triggered
 740  *     |    +------- Is a Wake GPE
 741  *     +------------ <Reserved>
 742  */
 743 #define ACPI_GPE_DISPATCH_NONE          (UINT8) 0x00
 744 #define ACPI_GPE_DISPATCH_METHOD        (UINT8) 0x01
 745 #define ACPI_GPE_DISPATCH_HANDLER       (UINT8) 0x02
 746 #define ACPI_GPE_DISPATCH_NOTIFY        (UINT8) 0x03
 747 #define ACPI_GPE_DISPATCH_MASK          (UINT8) 0x03

 749 #define ACPI_GPE_LEVEL_TRIGGERED        (UINT8) 0x04
 750 #define ACPI_GPE_EDGE_TRIGGERED         (UINT8) 0x00
 751 #define ACPI_GPE_XRUPT_TYPE_MASK        (UINT8) 0x04

 753 #define ACPI_GPE_CAN_WAKE               (UINT8) 0x08

 755 /*
 756  * Flags for GPE and Lock interfaces
 757  */
 758 #define ACPI_NOT_ISR                    0x1
 759 #define ACPI_ISR                        0x0


 762 /* Notify types */

 764 #define ACPI_SYSTEM_NOTIFY              0x1
 765 #define ACPI_DEVICE_NOTIFY              0x2
 766 #define ACPI_ALL_NOTIFY                 (ACPI_SYSTEM_NOTIFY | ACPI_DEVICE_NOTIFY
 767 #define ACPI_MAX_NOTIFY_HANDLER_TYPE    0x3
 768 #define ACPI_NUM_NOTIFY_TYPES           2

 770 #define ACPI_MAX_SYS_NOTIFY             0x7F
```

```
771 #define ACPI_MAX_DEVICE_SPECIFIC_NOTIFY 0xBF
704 #define ACPI_MAX_SYS_NOTIFY             0x7f

773 #define ACPI_SYSTEM_HANDLER_LIST        0 /* Used as index, must be SYSTEM_NOTIF
774 #define ACPI_DEVICE_HANDLER_LIST        1 /* Used as index, must be DEVICE_NOTIF


777 /* Address Space (Operation Region) Types */

779 typedef UINT8                           ACPI_ADR_SPACE_TYPE;

781 #define ACPI_ADR_SPACE_SYSTEM_MEMORY    (ACPI_ADR_SPACE_TYPE) 0
782 #define ACPI_ADR_SPACE_SYSTEM_IO        (ACPI_ADR_SPACE_TYPE) 1
783 #define ACPI_ADR_SPACE_PCI_CONFIG       (ACPI_ADR_SPACE_TYPE) 2
784 #define ACPI_ADR_SPACE_EC               (ACPI_ADR_SPACE_TYPE) 3
785 #define ACPI_ADR_SPACE_SMBUS            (ACPI_ADR_SPACE_TYPE) 4
786 #define ACPI_ADR_SPACE_CMOS             (ACPI_ADR_SPACE_TYPE) 5
787 #define ACPI_ADR_SPACE_PCI_BAR_TARGET   (ACPI_ADR_SPACE_TYPE) 6
788 #define ACPI_ADR_SPACE_IPMI             (ACPI_ADR_SPACE_TYPE) 7
789 #define ACPI_ADR_SPACE_GPIO             (ACPI_ADR_SPACE_TYPE) 8
790 #define ACPI_ADR_SPACE_GSBUS            (ACPI_ADR_SPACE_TYPE) 9
791 #define ACPI_ADR_SPACE_PLATFORM_COMM    (ACPI_ADR_SPACE_TYPE) 10

793 #define ACPI_NUM_PREDEFINED_REGIONS     11
720 #define ACPI_NUM_PREDEFINED_REGIONS     8

795 /*
796  * Special Address Spaces
797  *
798  * Note: A Data Table region is a special type of operation region
799  * that has its own AML opcode. However, internally, the AML
800  * interpreter simply creates an operation region with an an address
801  * space type of ACPI_ADR_SPACE_DATA_TABLE.
802  */
803 #define ACPI_ADR_SPACE_DATA_TABLE       (ACPI_ADR_SPACE_TYPE) 0x7E /* Internal t
804 #define ACPI_ADR_SPACE_FIXED_HARDWARE   (ACPI_ADR_SPACE_TYPE) 0x7F

806 /* Values for _REG connection code */

808 #define ACPI_REG_DISCONNECT             0
809 #define ACPI_REG_CONNECT                1

811 /*
812  * BitRegister IDs
813  *
814  * These values are intended to be used by the hardware interfaces
815  * and are mapped to individual bitfields defined within the ACPI
816  * registers. See the AcpiGbl_BitRegisterInfo global table in utglobal.c
817  * for this mapping.
818  */

820 /* PM1 Status register */

822 #define ACPI_BITREG_TIMER_STATUS                0x00
823 #define ACPI_BITREG_BUS_MASTER_STATUS           0x01
824 #define ACPI_BITREG_GLOBAL_LOCK_STATUS          0x02
825 #define ACPI_BITREG_POWER_BUTTON_STATUS         0x03
826 #define ACPI_BITREG_SLEEP_BUTTON_STATUS         0x04
827 #define ACPI_BITREG_RT_CLOCK_STATUS             0x05
828 #define ACPI_BITREG_WAKE_STATUS                 0x06
829 #define ACPI_BITREG_PCIEXP_WAKE_STATUS          0x07

831 /* PM1 Enable register */

833 #define ACPI_BITREG_TIMER_ENABLE                0x08
834 #define ACPI_BITREG_GLOBAL_LOCK_ENABLE          0x09
```



```
835 #define ACPI_BITREG_POWER_BUTTON_ENABLE         0x0A
836 #define ACPI_BITREG_SLEEP_BUTTON_ENABLE         0x0B
837 #define ACPI_BITREG_RT_CLOCK_ENABLE             0x0C
838 #define ACPI_BITREG_PCIEXP_WAKE_DISABLE         0x0D

840 /* PM1 Control register */

842 #define ACPI_BITREG_SCI_ENABLE                  0x0E
843 #define ACPI_BITREG_BUS_MASTER_RLD              0x0F
844 #define ACPI_BITREG_GLOBAL_LOCK_RELEASE         0x10
845 #define ACPI_BITREG_SLEEP_TYPE                  0x11
846 #define ACPI_BITREG_SLEEP_ENABLE                0x12

848 /* PM2 Control register */

850 #define ACPI_BITREG_ARB_DISABLE                 0x13

852 #define ACPI_BITREG_MAX                         0x13
853 #define ACPI_NUM_BITREG                         ACPI_BITREG_MAX + 1


856 /* Status register values. A 1 clears a status bit. 0 = no effect */

858 #define ACPI_CLEAR_STATUS                       1

860 /* Enable and Control register values */

862 #define ACPI_ENABLE_EVENT                       1
863 #define ACPI_DISABLE_EVENT                      0


866 /* Sleep function dispatch */

868 typedef ACPI_STATUS (*ACPI_SLEEP_FUNCTION) (
869     UINT8                   SleepState);

871 typedef struct acpi_sleep_functions
872 {
873     ACPI_SLEEP_FUNCTION     LegacyFunction;
874     ACPI_SLEEP_FUNCTION     ExtendedFunction;

876 } ACPI_SLEEP_FUNCTIONS;


879 /*
880  * External ACPI object definition
881  */

883 /*
884  * Note: Type == ACPI_TYPE_ANY (0) is used to indicate a NULL package element
885  * or an unresolved named reference.
886  */
887 typedef union acpi_object
888 {
889     ACPI_OBJECT_TYPE                Type;   /* See definition of AcpiNsType for
890     struct
891     {
892         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_INTEGER */
893         UINT64                          Value;      /* The actual number */
894     } Integer;

896     struct
897     {
898         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_STRING */
899         UINT32                          Length;     /* # of bytes in string, exc
900         char                            *Pointer;   /* points to the string valu
```

```
 901     } String;

 903     struct
 904     {
 905         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_BUFFER */
 906         UINT32                          Length;     /* # of bytes in buffer */
 907         UINT8                           *Pointer;   /* points to the buffer */
 908     } Buffer;

 910     struct
 911     {
 912         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_PACKAGE */
 913         UINT32                          Count;      /* # of elements in package
 914         union acpi_object               *Elements;  /* Pointer to an array of AC
 915     } Package;

 917     struct
 918     {
 919         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_LOCAL_REFERENCE
 920         ACPI_OBJECT_TYPE                ActualType; /* Type associated with the
 921         ACPI_HANDLE                     Handle;     /* object reference */
 922     } Reference;

 924     struct
 925     {
 926         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_PROCESSOR */
 927         UINT32                          ProcId;
 928         ACPI_IO_ADDRESS                 PblkAddress;
 929         UINT32                          PblkLength;
 930     } Processor;

 932     struct
 933     {
 934         ACPI_OBJECT_TYPE                Type;       /* ACPI_TYPE_POWER */
 935         UINT32                          SystemLevel;
 936         UINT32                          ResourceOrder;
 937     } PowerResource;

 939 } ACPI_OBJECT;
______unchanged_portion_omitted_


 953 /*
 954  * Miscellaneous common Data Structures used by the interfaces
 955  */
 956 #define ACPI_NO_BUFFER              0
 957 #define ACPI_ALLOCATE_BUFFER        (ACPI_SIZE) (-1)    /* Let ACPICA allocate b
 958 #define ACPI_ALLOCATE_LOCAL_BUFFER  (ACPI_SIZE) (-2)    /* For internal use only
 871 #define ACPI_ALLOCATE_BUFFER        (ACPI_SIZE) (-1)
 872 #define ACPI_ALLOCATE_LOCAL_BUFFER  (ACPI_SIZE) (-2)

 960 typedef struct acpi_buffer
 961 {
 962     ACPI_SIZE                       Length;         /* Length in bytes of the bu
 963     void                            *Pointer;       /* pointer to buffer */

 965 } ACPI_BUFFER;
______unchanged_portion_omitted_


1026 /* Table Event Types */

1028 #define ACPI_TABLE_EVENT_LOAD           0x0
1029 #define ACPI_TABLE_EVENT_UNLOAD         0x1
1030 #define ACPI_NUM_TABLE_EVENTS           2
```



```
1033 /*
1034  * Types specific to the OS service interfaces
1035  */
1036 typedef UINT32
1037 (ACPI_SYSTEM_XFACE *ACPI_OSD_HANDLER) (
1038     void                            *Context);

1040 typedef void
1041 (ACPI_SYSTEM_XFACE *ACPI_OSD_EXEC_CALLBACK) (
1042     void                            *Context);

1044 /*
1045  * Various handlers and callback procedures
1046  */
1047 typedef
1048 UINT32 (*ACPI_SCI_HANDLER) (
1049     void                            *Context);

1051 typedef
1052 void (*ACPI_GBL_EVENT_HANDLER) (
1053     UINT32                          EventType,
1054     ACPI_HANDLE                     Device,
1055     UINT32                          EventNumber,
1056     void                            *Context);

1058 #define ACPI_EVENT_TYPE_GPE         0
1059 #define ACPI_EVENT_TYPE_FIXED       1

1061 typedef
1062 UINT32 (*ACPI_EVENT_HANDLER) (
1063     void                            *Context);

1065 typedef
1066 UINT32 (*ACPI_GPE_HANDLER) (
1067     ACPI_HANDLE                     GpeDevice,
1068     UINT32                          GpeNumber,
1069     void                            *Context);

1071 typedef
1072 void (*ACPI_NOTIFY_HANDLER) (
1073     ACPI_HANDLE                     Device,
1074     UINT32                          Value,
1075     void                            *Context);

1077 typedef
1078 void (*ACPI_OBJECT_HANDLER) (
1079     ACPI_HANDLE                     Object,
1080     void                            *Data);

1082 typedef
1083 ACPI_STATUS (*ACPI_INIT_HANDLER) (
1084     ACPI_HANDLE                     Object,
1085     UINT32                          Function);

1087 #define ACPI_INIT_DEVICE_INI        1

1089 typedef
1090 ACPI_STATUS (*ACPI_EXCEPTION_HANDLER) (
1091     ACPI_STATUS                     AmlStatus,
1092     ACPI_NAME                       Name,
1093     UINT16                          Opcode,
1094     UINT32                          AmlOffset,
1095     void                            *Context);

1097 /* Table Event handler (Load, LoadTable, etc.) and types */
```

```
typedef
ACPI_STATUS (*ACPI_TABLE_HANDLER) (
    UINT32                          Event,
    void                            *Table,
    void                            *Context);

#define ACPI_TABLE_LOAD             0x0
#define ACPI_TABLE_UNLOAD           0x1
#define ACPI_NUM_TABLE_EVENTS       2


/* Address Spaces (For Operation Regions) */

typedef
ACPI_STATUS (*ACPI_ADR_SPACE_HANDLER) (
    UINT32                          Function,
    ACPI_PHYSICAL_ADDRESS           Address,
    UINT32                          BitWidth,
    UINT64                          *Value,
    void                            *HandlerContext,
    void                            *RegionContext);

#define ACPI_DEFAULT_HANDLER            NULL

/* Special Context data for GenericSerialBus/GeneralPurposeIo (ACPI 5.0) */

typedef struct acpi_connection_info
{
    UINT8                           *Connection;
    UINT16                          Length;
    UINT8                           AccessLength;

} ACPI_CONNECTION_INFO;


typedef
ACPI_STATUS (*ACPI_ADR_SPACE_SETUP) (
    ACPI_HANDLE                     RegionHandle,
    UINT32                          Function,
    void                            *HandlerContext,
    void                            **RegionContext);

#define ACPI_REGION_ACTIVATE    0
#define ACPI_REGION_DEACTIVATE  1

typedef
ACPI_STATUS (*ACPI_WALK_CALLBACK) (
    ACPI_HANDLE                     Object,
    UINT32                          NestingLevel,
    void                            *Context,
    void                            **ReturnValue);

typedef
UINT32 (*ACPI_INTERFACE_HANDLER) (
    ACPI_STRING                     InterfaceName,
    UINT32                          Supported);


/* Interrupt handler return values */

#define ACPI_INTERRUPT_NOT_HANDLED      0x00
#define ACPI_INTERRUPT_HANDLED          0x01

/* GPE handler return values */
```



```
#define ACPI_REENABLE_GPE               0x80


/* Length of 32-bit EISAID values when converted back to a string */

#define ACPI_EISAID_STRING_SIZE         8   /* Includes null terminator */

/* Length of UUID (string) values */

#define ACPI_UUID_LENGTH                16


/* Structures used for device/processor HID, UID, CID, and SUB */
/* Structures used for device/processor HID, UID, CID */

typedef struct acpi_pnp_device_id
typedef struct acpi_device_id
{
    UINT32                          Length;             /* Length of string + nu
    char                            *String;

} ACPI_PNP_DEVICE_ID;
} ACPI_DEVICE_ID;

typedef struct acpi_pnp_device_id_list
typedef struct acpi_device_id_list
{
    UINT32                          Count;              /* Number of IDs in Ids
    UINT32                          ListSize;           /* Size of list, includi
    ACPI_PNP_DEVICE_ID              Ids[1];             /* ID array */
    ACPI_DEVICE_ID                  Ids[1];             /* ID array */

} ACPI_PNP_DEVICE_ID_LIST;
} ACPI_DEVICE_ID_LIST;

/*
 * Structure returned from AcpiGetObjectInfo.
 * Optimized for both 32- and 64-bit builds
 */
typedef struct acpi_device_info
{
    UINT32                          InfoSize;           /* Size of info, includi
    UINT32                          Name;               /* ACPI object Name */
    ACPI_OBJECT_TYPE                Type;               /* ACPI object Type */
    UINT8                           ParamCount;         /* If a method, required
    UINT8                           Valid;              /* Indicates which optio
    UINT8                           Flags;              /* Miscellaneous info */
    UINT8                           HighestDstates[4];  /* _SxD values: 0xFF ind
    UINT8                           LowestDstates[5];   /* _SxW values: 0xFF ind
    UINT32                          CurrentStatus;      /* _STA value */
    UINT64                          Address;            /* _ADR value */
    ACPI_PNP_DEVICE_ID              HardwareId;         /* _HID value */
    ACPI_PNP_DEVICE_ID              UniqueId;           /* _UID value */
    ACPI_PNP_DEVICE_ID              SubsystemId;        /* _SUB value */
    ACPI_PNP_DEVICE_ID_LIST         CompatibleIdList;   /* _CID list <must be la
    ACPI_DEVICE_ID                  HardwareId;         /* _HID value */
    ACPI_DEVICE_ID                  UniqueId;           /* _UID value */
    ACPI_DEVICE_ID_LIST             CompatibleIdList;   /* _CID list <must be la

} ACPI_DEVICE_INFO;

/* Values for Flags field above (AcpiGetObjectInfo) */

#define ACPI_PCI_ROOT_BRIDGE            0x01

/* Flags for Valid field above (AcpiGetObjectInfo) */
```

```
1222 #define ACPI_VALID_STA                  0x01
1223 #define ACPI_VALID_ADR                  0x02
1224 #define ACPI_VALID_HID                  0x04
1225 #define ACPI_VALID_UID                  0x08
1226 #define ACPI_VALID_SUB                  0x10
1227 #define ACPI_VALID_CID                  0x20
1228 #define ACPI_VALID_SXDS                 0x40
1229 #define ACPI_VALID_SXWS                 0x80
1124 #define ACPI_VALID_CID                  0x10
1125 #define ACPI_VALID_SXDS                 0x20
1126 #define ACPI_VALID_SXWS                 0x40

1231 /* Flags for _STA return value (CurrentStatus above) */
1128 /* Flags for _STA method */

1233 #define ACPI_STA_DEVICE_PRESENT         0x01
1234 #define ACPI_STA_DEVICE_ENABLED         0x02
1235 #define ACPI_STA_DEVICE_UI              0x04
1236 #define ACPI_STA_DEVICE_FUNCTIONING     0x08
1237 #define ACPI_STA_DEVICE_OK              0x08 /* Synonym */
1238 #define ACPI_STA_BATTERY_PRESENT        0x10


1241 /* Context structs for address space handlers */

1243 typedef struct acpi_pci_id
1244 {
1245     UINT16                          Segment;
1246     UINT16                          Bus;
1247     UINT16                          Device;
1248     UINT16                          Function;

1250 } ACPI_PCI_ID;
______unchanged_portion_omitted_


1263 /*
1264  * ACPI_MEMORY_LIST is used only if the ACPICA local cache is enabled
1265  */
1266 typedef struct acpi_memory_list
1267 {
1268     char                            *ListName;
1269     void                            *ListHead;
1270     UINT16                          ObjectSize;
1271     UINT16                          MaxDepth;
1272     UINT16                          CurrentDepth;
1170     UINT16                          LinkOffset;

1274 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

1276     /* Statistics for debug memory tracking only */

1278     UINT32                          TotalAllocated;
1279     UINT32                          TotalFreed;
1280     UINT32                          MaxOccupied;
1281     UINT32                          TotalSize;
1282     UINT32                          CurrentTotalSize;
1283     UINT32                          Requests;
1284     UINT32                          Hits;
1285 #endif

1287 } ACPI_MEMORY_LIST;


1290 /* Definitions of _OSI support */
```



```
1292 #define ACPI_VENDOR_STRINGS                 0x01
1293 #define ACPI_FEATURE_STRINGS                0x02
1294 #define ACPI_ENABLE_INTERFACES              0x00
1295 #define ACPI_DISABLE_INTERFACES             0x04

1297 #define ACPI_DISABLE_ALL_VENDOR_STRINGS     (ACPI_DISABLE_INTERFACES | ACPI_VEND
1298 #define ACPI_DISABLE_ALL_FEATURE_STRINGS    (ACPI_DISABLE_INTERFACES | ACPI_FEAT
1299 #define ACPI_DISABLE_ALL_STRINGS            (ACPI_DISABLE_INTERFACES | ACPI_VEND
1300 #define ACPI_ENABLE_ALL_VENDOR_STRINGS      (ACPI_ENABLE_INTERFACES | ACPI_VENDO
1301 #define ACPI_ENABLE_ALL_FEATURE_STRINGS     (ACPI_ENABLE_INTERFACES | ACPI_FEATU
1302 #define ACPI_ENABLE_ALL_STRINGS             (ACPI_ENABLE_INTERFACES | ACPI_VENDO

1304 #define ACPI_OSI_WIN_2000               0x01
1305 #define ACPI_OSI_WIN_XP                 0x02
1306 #define ACPI_OSI_WIN_XP_SP1             0x03
1307 #define ACPI_OSI_WINSRV_2003            0x04
1308 #define ACPI_OSI_WIN_XP_SP2             0x05
1309 #define ACPI_OSI_WINSRV_2003_SP1        0x06
1310 #define ACPI_OSI_WIN_VISTA              0x07
1311 #define ACPI_OSI_WINSRV_2008            0x08
1312 #define ACPI_OSI_WIN_VISTA_SP1          0x09
1313 #define ACPI_OSI_WIN_VISTA_SP2          0x0A
1314 #define ACPI_OSI_WIN_7                  0x0B
1315 #define ACPI_OSI_WIN_8                  0x0C


1318 #endif /* __ACTYPES_H__ */
```

```
**********************************************************
   27274 Thu Dec 26 13:49:58 2013
new/usr/src/common/acpica/include/acutils.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acutils.h -- prototypes for the common (subsystem-wide) procedures
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _ACUTILS_H
#define _ACUTILS_H


extern const UINT8                      AcpiGbl_ResourceAmlSizes[];
extern const UINT8                      AcpiGbl_ResourceAmlSerialBusSizes[];

/* Strings used by the disassembler and debugger resource dump routines */

#if defined(ACPI_DISASSEMBLER) || defined (ACPI_DEBUGGER)

extern const char                       *AcpiGbl_BmDecode[];
extern const char                       *AcpiGbl_ConfigDecode[];
extern const char                       *AcpiGbl_ConsumeDecode[];
extern const char                       *AcpiGbl_DecDecode[];
```



```
extern const char                       *AcpiGbl_HeDecode[];
extern const char                       *AcpiGbl_IoDecode[];
extern const char                       *AcpiGbl_LlDecode[];
extern const char                       *AcpiGbl_MaxDecode[];
extern const char                       *AcpiGbl_MemDecode[];
extern const char                       *AcpiGbl_MinDecode[];
extern const char                       *AcpiGbl_MtpDecode[];
extern const char                       *AcpiGbl_RngDecode[];
extern const char                       *AcpiGbl_RwDecode[];
extern const char                       *AcpiGbl_ShrDecode[];
extern const char                       *AcpiGbl_SizDecode[];
extern const char                       *AcpiGbl_TrsDecode[];
extern const char                       *AcpiGbl_TtpDecode[];
extern const char                       *AcpiGbl_TypDecode[];
extern const char                       *AcpiGbl_PpcDecode[];
extern const char                       *AcpiGbl_IorDecode[];
extern const char                       *AcpiGbl_DtsDecode[];
extern const char                       *AcpiGbl_CtDecode[];
extern const char                       *AcpiGbl_SbtDecode[];
extern const char                       *AcpiGbl_AmDecode[];
extern const char                       *AcpiGbl_SmDecode[];
extern const char                       *AcpiGbl_WmDecode[];
extern const char                       *AcpiGbl_CphDecode[];
extern const char                       *AcpiGbl_CpoDecode[];
extern const char                       *AcpiGbl_DpDecode[];
extern const char                       *AcpiGbl_EdDecode[];
extern const char                       *AcpiGbl_BpbDecode[];
extern const char                       *AcpiGbl_SbDecode[];
extern const char                       *AcpiGbl_FcDecode[];
extern const char                       *AcpiGbl_PtDecode[];
#endif

/*
 * For the iASL compiler case, the output is redirected to stderr so that
 * any of the various ACPI errors and warnings do not appear in the output
 * files, for either the compiler or disassembler portions of the tool.
 */
#ifdef ACPI_ASL_COMPILER

#include <stdio.h>
extern FILE                 *AcpiGbl_OutputFile;

#define ACPI_MSG_REDIRECT_BEGIN \
    FILE                    *OutputFile = AcpiGbl_OutputFile; \
    AcpiOsRedirectOutput (stderr);

#define ACPI_MSG_REDIRECT_END \
    AcpiOsRedirectOutput (OutputFile);

#else
/*
 * non-iASL case - no redirection, nothing to do
 */
#define ACPI_MSG_REDIRECT_BEGIN
#define ACPI_MSG_REDIRECT_END
#endif

/*
 * Common error message prefixes
 */
#define ACPI_MSG_ERROR          "ACPI Error: "
#define ACPI_MSG_EXCEPTION      "ACPI Exception: "
#define ACPI_MSG_WARNING        "ACPI Warning: "
#define ACPI_MSG_INFO           "ACPI: "

#define ACPI_MSG_BIOS_ERROR     "ACPI BIOS Error (bug): "
```

```
125 #define ACPI_MSG_BIOS_WARNING   "ACPI BIOS Warning (bug): "

127 /*
128  * Common message suffix
129  */
130 #define ACPI_MSG_SUFFIX \
131     AcpiOsPrintf (" (%8.8X/%s-%u)\n", ACPI_CA_VERSION, ModuleName, LineNumber)


134 /* Types for Resource descriptor entries */

136 #define ACPI_INVALID_RESOURCE           0
137 #define ACPI_FIXED_LENGTH               1
138 #define ACPI_VARIABLE_LENGTH            2
139 #define ACPI_SMALL_VARIABLE_LENGTH      3

141 typedef
142 ACPI_STATUS (*ACPI_WALK_AML_CALLBACK) (
143     UINT8                   *Aml,
144     UINT32                  Length,
145     UINT32                  Offset,
146     UINT8                   ResourceIndex,
147     void                    **Context);
 87     void                    *Context);

149 typedef
150 ACPI_STATUS (*ACPI_PKG_CALLBACK) (
151     UINT8                   ObjectType,
152     ACPI_OPERAND_OBJECT     *SourceObject,
153     ACPI_GENERIC_STATE      *State,
154     void                    *Context);

156 typedef struct acpi_pkg_info
157 {
158     UINT8                   *FreeSpace;
159     ACPI_SIZE               Length;
160     UINT32                  ObjectSpace;
161     UINT32                  NumPackages;

163 } ACPI_PKG_INFO;

165 /* Object reference counts */

167 #define REF_INCREMENT       (UINT16) 0
168 #define REF_DECREMENT       (UINT16) 1
107 #define REF_FORCE_DELETE    (UINT16) 2

170 /* AcpiUtDumpBuffer */

172 #define DB_BYTE_DISPLAY     1
173 #define DB_WORD_DISPLAY     2
174 #define DB_DWORD_DISPLAY    4
175 #define DB_QWORD_DISPLAY    8


177 /*
178  * utglobal - Global data structures and procedures
179  */
180 ACPI_STATUS
181 AcpiUtInitGlobals (
182     void);

184 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

186 char *
187 AcpiUtGetMutexName (
```



```
188     UINT32                  MutexId);

190 const char *
191 AcpiUtGetNotifyName (
192     UINT32                  NotifyValue);

194 #endif

196 char *
197 AcpiUtGetTypeName (
198     ACPI_OBJECT_TYPE        Type);

200 char *
201 AcpiUtGetNodeName (
202     void                    *Object);

204 char *
205 AcpiUtGetDescriptorName (
206     void                    *Object);

208 const char *
209 AcpiUtGetReferenceName (
210     ACPI_OPERAND_OBJECT     *Object);

212 char *
213 AcpiUtGetObjectTypeName (
214     ACPI_OPERAND_OBJECT     *ObjDesc);

216 char *
217 AcpiUtGetRegionName (
218     UINT8                   SpaceId);

220 char *
221 AcpiUtGetEventName (
222     UINT32                  EventId);

224 char
225 AcpiUtHexToAsciiChar (
226     UINT64                  Integer,
227     UINT32                  Position);

229 BOOLEAN
230 AcpiUtValidObjectType (
231     ACPI_OBJECT_TYPE        Type);


234 /*
235  * utinit - miscellaneous initialization and shutdown
236  */
237 ACPI_STATUS
238 AcpiUtHardwareInitialize (
239     void);

241 void
242 AcpiUtSubsystemShutdown (
243     void);


246 /*
247  * utclib - Local implementations of C library functions
248  */
249 #ifndef ACPI_USE_SYSTEM_CLIBRARY

251 ACPI_SIZE
252 AcpiUtStrlen (
253     const char              *String);
```

```
255 char *
256 AcpiUtStrcpy (
257     char                    *DstString,
258     const char              *SrcString);

260 char *
261 AcpiUtStrncpy (
262     char                    *DstString,
263     const char              *SrcString,
264     ACPI_SIZE               Count);

266 int
267 AcpiUtMemcmp (
268     const char              *Buffer1,
269     const char              *Buffer2,
270     ACPI_SIZE               Count);

272 int
273 AcpiUtStrncmp (
274     const char              *String1,
275     const char              *String2,
276     ACPI_SIZE               Count);

278 int
279 AcpiUtStrcmp (
280     const char              *String1,
281     const char              *String2);

283 char *
284 AcpiUtStrcat (
285     char                    *DstString,
286     const char              *SrcString);

288 char *
289 AcpiUtStrncat (
290     char                    *DstString,
291     const char              *SrcString,
292     ACPI_SIZE               Count);

294 UINT32
295 AcpiUtStrtoul (
296     const char              *String,
297     char                    **Terminator,
298     UINT32                  Base);

300 char *
301 AcpiUtStrstr (
302     char                    *String1,
303     char                    *String2);

305 void *
306 AcpiUtMemcpy (
307     void                    *Dest,
308     const void              *Src,
309     ACPI_SIZE               Count);

311 void *
312 AcpiUtMemset (
313     void                    *Dest,
314     UINT8                   Value,
315     ACPI_SIZE               Count);

317 int
318 AcpiUtToUpper (
319     int                     c);
```



```
321 int
322 AcpiUtToLower (
323     int                     c);

325 extern const UINT8 _acpi_ctype[];

327 #define _ACPI_XA     0x00    /* extra alphabetic - not supported */
328 #define _ACPI_XS     0x40    /* extra space */
329 #define _ACPI_BB     0x00    /* BEL, BS, etc. - not supported */
330 #define _ACPI_CN     0x20    /* CR, FF, HT, NL, VT */
331 #define _ACPI_DI     0x04    /* '0'-'9' */
332 #define _ACPI_LO     0x02    /* 'a'-'z' */
333 #define _ACPI_PU     0x10    /* punctuation */
334 #define _ACPI_SP     0x08    /* space */
335 #define _ACPI_UP     0x01    /* 'A'-'Z' */
336 #define _ACPI_XD     0x80    /* '0'-'9', 'A'-'F', 'a'-'f' */

338 #define ACPI_IS_DIGIT(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_DI))
339 #define ACPI_IS_SPACE(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_SP))
340 #define ACPI_IS_XDIGIT(c) (_acpi_ctype[(unsigned char)(c)] & (_ACPI_XD))
341 #define ACPI_IS_UPPER(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_UP))
342 #define ACPI_IS_LOWER(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_LO))
343 #define ACPI_IS_PRINT(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_LO | _ACPI_U
344 #define ACPI_IS_ALPHA(c)  (_acpi_ctype[(unsigned char)(c)] & (_ACPI_LO | _ACPI_U

346 #endif /* !ACPI_USE_SYSTEM_CLIBRARY */

348 #define ACPI_IS_ASCII(c)  ((c) < 0x80)


351 /*
352  * utcopy - Object construction and conversion interfaces
353  */
354 ACPI_STATUS
355 AcpiUtBuildSimpleObject(
356     ACPI_OPERAND_OBJECT     *Obj,
357     ACPI_OBJECT             *UserObj,
358     UINT8                   *DataSpace,
359     UINT32                  *BufferSpaceUsed);

361 ACPI_STATUS
362 AcpiUtBuildPackageObject (
363     ACPI_OPERAND_OBJECT     *Obj,
364     UINT8                   *Buffer,
365     UINT32                  *SpaceUsed);

367 ACPI_STATUS
368 AcpiUtCopyIobjectToEobject (
369     ACPI_OPERAND_OBJECT     *Obj,
370     ACPI_BUFFER             *RetBuffer);

372 ACPI_STATUS
373 AcpiUtCopyEobjectToIobject (
374     ACPI_OBJECT             *Obj,
375     ACPI_OPERAND_OBJECT     **InternalObj);

377 ACPI_STATUS
378 AcpiUtCopyISimpleToIsimple (
379     ACPI_OPERAND_OBJECT     *SourceObj,
380     ACPI_OPERAND_OBJECT     *DestObj);

382 ACPI_STATUS
383 AcpiUtCopyIobjectToIobject (
384     ACPI_OPERAND_OBJECT     *SourceDesc,
385     ACPI_OPERAND_OBJECT     **DestDesc,
```

```
386     ACPI_WALK_STATE         *WalkState);


389 /*
390  * utcreate - Object creation
391  */
392 ACPI_STATUS
393 AcpiUtUpdateObjectReference (
394     ACPI_OPERAND_OBJECT     *Object,
395     UINT16                  Action);


398 /*
399  * utdebug - Debug interfaces
400  */
401 void
402 AcpiUtInitStackPtrTrace (
403     void);

405 void
406 AcpiUtTrackStackPtr (
407     void);

409 void
410 AcpiUtTrace (
411     UINT32                  LineNumber,
412     const char              *FunctionName,
413     const char              *ModuleName,
414     UINT32                  ComponentId);

416 void
417 AcpiUtTracePtr (
418     UINT32                  LineNumber,
419     const char              *FunctionName,
420     const char              *ModuleName,
421     UINT32                  ComponentId,
422     void                    *Pointer);

424 void
425 AcpiUtTraceU32 (
426     UINT32                  LineNumber,
427     const char              *FunctionName,
428     const char              *ModuleName,
429     UINT32                  ComponentId,
430     UINT32                  Integer);

432 void
433 AcpiUtTraceStr (
434     UINT32                  LineNumber,
435     const char              *FunctionName,
436     const char              *ModuleName,
437     UINT32                  ComponentId,
438     char                    *String);

440 void
441 AcpiUtExit (
442     UINT32                  LineNumber,
443     const char              *FunctionName,
444     const char              *ModuleName,
445     UINT32                  ComponentId);

447 void
448 AcpiUtStatusExit (
449     UINT32                  LineNumber,
450     const char              *FunctionName,
451     const char              *ModuleName,
```



```
452     UINT32                  ComponentId,
453     ACPI_STATUS             Status);

455 void
456 AcpiUtValueExit (
457     UINT32                  LineNumber,
458     const char              *FunctionName,
459     const char              *ModuleName,
460     UINT32                  ComponentId,
461     UINT64                  Value);

463 void
464 AcpiUtPtrExit (
465     UINT32                  LineNumber,
466     const char              *FunctionName,
467     const char              *ModuleName,
468     UINT32                  ComponentId,
469     UINT8                   *Ptr);

471 void
472 AcpiUtDebugDumpBuffer (
412 AcpiUtDumpBuffer (
473     UINT8                   *Buffer,
474     UINT32                  Count,
475     UINT32                  Display,
476     UINT32                  ComponentId);
416     UINT32                  componentId);

478 void
479 AcpiUtDumpBuffer (
419 AcpiUtDumpBuffer2 (
480     UINT8                   *Buffer,
481     UINT32                  Count,
482     UINT32                  Display,
483     UINT32                  Offset);
422     UINT32                  Display);

485 void
486 AcpiUtReportError (
487     char                    *ModuleName,
488     UINT32                  LineNumber);

490 void
491 AcpiUtReportInfo (
492     char                    *ModuleName,
493     UINT32                  LineNumber);

495 void
496 AcpiUtReportWarning (
497     char                    *ModuleName,
498     UINT32                  LineNumber);

500 /*
501  * utdelete - Object deletion and reference counts
502  */
503 void
504 AcpiUtAddReference (
505     ACPI_OPERAND_OBJECT     *Object);

507 void
508 AcpiUtRemoveReference (
509     ACPI_OPERAND_OBJECT     *Object);

511 void
512 AcpiUtDeleteInternalPackageObject (
513     ACPI_OPERAND_OBJECT     *Object);
```

```
515 void
516 AcpiUtDeleteInternalSimpleObject (
517     ACPI_OPERAND_OBJECT     *Object);

519 void
520 AcpiUtDeleteInternalObjectList (
521     ACPI_OPERAND_OBJECT     **ObjList);


524 /*
525  * uteval - object evaluation
526  */
527 ACPI_STATUS
528 AcpiUtEvaluateObject (
529     ACPI_NAMESPACE_NODE     *PrefixNode,
530     char                    *Path,
531     UINT32                  ExpectedReturnBtypes,
532     ACPI_OPERAND_OBJECT     **ReturnDesc);

534 ACPI_STATUS
535 AcpiUtEvaluateNumericObject (
536     char                    *ObjectName,
537     ACPI_NAMESPACE_NODE     *DeviceNode,
538     UINT64                  *Value);

540 ACPI_STATUS
541 AcpiUtExecute_STA (
542     ACPI_NAMESPACE_NODE     *DeviceNode,
543     UINT32                  *StatusFlags);

545 ACPI_STATUS
546 AcpiUtExecutePowerMethods (
547     ACPI_NAMESPACE_NODE     *DeviceNode,
548     const char              **MethodNames,
549     UINT8                   MethodCount,
550     UINT8                   *OutValues);


553 /*
554  * utids - device ID support
555  */
556 ACPI_STATUS
557 AcpiUtExecute_HID (
558     ACPI_NAMESPACE_NODE     *DeviceNode,
559     ACPI_PNP_DEVICE_ID      **ReturnId);
498     ACPI_DEVICE_ID          **ReturnId);

561 ACPI_STATUS
562 AcpiUtExecute_UID (
563     ACPI_NAMESPACE_NODE     *DeviceNode,
564     ACPI_PNP_DEVICE_ID      **ReturnId);
503     ACPI_DEVICE_ID          **ReturnId);

566 ACPI_STATUS
567 AcpiUtExecute_SUB (
568     ACPI_NAMESPACE_NODE     *DeviceNode,
569     ACPI_PNP_DEVICE_ID      **ReturnId);

571 ACPI_STATUS
572 AcpiUtExecute_CID (
573     ACPI_NAMESPACE_NODE     *DeviceNode,
574     ACPI_PNP_DEVICE_ID_LIST **ReturnCidList);
508     ACPI_DEVICE_ID_LIST     **ReturnCidList);
```



```
577 /*
578  * utlock - reader/writer locks
579  */
580 ACPI_STATUS
581 AcpiUtCreateRwLock (
582     ACPI_RW_LOCK            *Lock);

584 void
585 AcpiUtDeleteRwLock (
586     ACPI_RW_LOCK            *Lock);

588 ACPI_STATUS
589 AcpiUtAcquireReadLock (
590     ACPI_RW_LOCK            *Lock);

592 ACPI_STATUS
593 AcpiUtReleaseReadLock (
594     ACPI_RW_LOCK            *Lock);

596 ACPI_STATUS
597 AcpiUtAcquireWriteLock (
598     ACPI_RW_LOCK            *Lock);

600 void
601 AcpiUtReleaseWriteLock (
602     ACPI_RW_LOCK            *Lock);


605 /*
606  * utobject - internal object create/delete/cache routines
607  */
608 ACPI_OPERAND_OBJECT  *
609 AcpiUtCreateInternalObjectDbg (
610     const char              *ModuleName,
611     UINT32                  LineNumber,
612     UINT32                  ComponentId,
613     ACPI_OBJECT_TYPE        Type);

615 void *
616 AcpiUtAllocateObjectDescDbg (
617     const char              *ModuleName,
618     UINT32                  LineNumber,
619     UINT32                  ComponentId);

621 #define AcpiUtCreateInternalObject(t)   AcpiUtCreateInternalObjectDbg (_AcpiModu
622 #define AcpiUtAllocateObjectDesc()      AcpiUtAllocateObjectDescDbg (_AcpiModule

624 void
625 AcpiUtDeleteObjectDesc (
626     ACPI_OPERAND_OBJECT     *Object);

628 BOOLEAN
629 AcpiUtValidInternalObject (
630     void                    *Object);

632 ACPI_OPERAND_OBJECT *
633 AcpiUtCreatePackageObject (
634     UINT32                  Count);

636 ACPI_OPERAND_OBJECT *
637 AcpiUtCreateIntegerObject (
638     UINT64                  Value);

640 ACPI_OPERAND_OBJECT *
641 AcpiUtCreateBufferObject (
642     ACPI_SIZE               BufferSize);
```

```
644 ACPI_OPERAND_OBJECT *
645 AcpiUtCreateStringObject (
646     ACPI_SIZE               StringSize);

648 ACPI_STATUS
649 AcpiUtGetObjectSize(
650     ACPI_OPERAND_OBJECT     *Obj,
651     ACPI_SIZE               *ObjLength);


654 /*
655  * utosi - Support for the _OSI predefined control method
656  */
657 ACPI_STATUS
658 AcpiUtInitializeInterfaces (
659     void);

661 ACPI_STATUS
595 void
662 AcpiUtInterfaceTerminate (
663     void);

665 ACPI_STATUS
666 AcpiUtInstallInterface (
667     ACPI_STRING             InterfaceName);

669 ACPI_STATUS
670 AcpiUtRemoveInterface (
671     ACPI_STRING             InterfaceName);

673 ACPI_STATUS
674 AcpiUtUpdateInterfaces (
675     UINT8                   Action);

677 ACPI_INTERFACE_INFO *
678 AcpiUtGetInterface (
679     ACPI_STRING             InterfaceName);

681 ACPI_STATUS
682 AcpiUtOsiImplementation (
683     ACPI_WALK_STATE         *WalkState);


686 /*
687  * utpredef - support for predefined names
688  */
689 const ACPI_PREDEFINED_INFO *
690 AcpiUtGetNextPredefinedMethod (
691     const ACPI_PREDEFINED_INFO  *ThisName);

693 const ACPI_PREDEFINED_INFO *
694 AcpiUtMatchPredefinedMethod (
695     char                        *Name);

697 const ACPI_PREDEFINED_INFO *
698 AcpiUtMatchResourceName (
699     char                        *Name);

701 void
702 AcpiUtDisplayPredefinedMethod (
703     char                        *Buffer,
704     const ACPI_PREDEFINED_INFO  *ThisName,
705     BOOLEAN                     MultiLine);

707 void
```



```
708 AcpiUtGetExpectedReturnTypes (
709     char                    *Buffer,
710     UINT32                  ExpectedBtypes);

712 UINT32
713 AcpiUtGetResourceBitWidth (
714     char                    *Buffer,
715     UINT16                  Types);


718 /*
719  * utstate - Generic state creation/cache routines
720  */
721 void
722 AcpiUtPushGenericState (
723     ACPI_GENERIC_STATE      **ListHead,
724     ACPI_GENERIC_STATE      *State);

726 ACPI_GENERIC_STATE *
727 AcpiUtPopGenericState (
728     ACPI_GENERIC_STATE      **ListHead);


731 ACPI_GENERIC_STATE *
732 AcpiUtCreateGenericState (
733     void);

735 ACPI_THREAD_STATE *
736 AcpiUtCreateThreadState (
737     void);

739 ACPI_GENERIC_STATE *
740 AcpiUtCreateUpdateState (
741     ACPI_OPERAND_OBJECT     *Object,
742     UINT16                  Action);

744 ACPI_GENERIC_STATE *
745 AcpiUtCreatePkgState (
746     void                    *InternalObject,
747     void                    *ExternalObject,
748     UINT16                  Index);

750 ACPI_STATUS
751 AcpiUtCreateUpdateStateAndPush (
752     ACPI_OPERAND_OBJECT     *Object,
753     UINT16                  Action,
754     ACPI_GENERIC_STATE      **StateList);

756 ACPI_STATUS
757 AcpiUtCreatePkgStateAndPush (
758     void                    *InternalObject,
759     void                    *ExternalObject,
760     UINT16                  Index,
761     ACPI_GENERIC_STATE      **StateList);

763 ACPI_GENERIC_STATE *
764 AcpiUtCreateControlState (
765     void);

767 void
768 AcpiUtDeleteGenericState (
769     ACPI_GENERIC_STATE      *State);


772 /*
773  * utmath
```

```
 */
ACPI_STATUS
AcpiUtDivide (
    UINT64                  InDividend,
    UINT64                  InDivisor,
    UINT64                  *OutQuotient,
    UINT64                  *OutRemainder);

ACPI_STATUS
AcpiUtShortDivide (
    UINT64                  InDividend,
    UINT64                  Divisor,
    UINT64                  *OutQuotient,
    UINT64                  *OutRemainder);


/*
 * utmisc
 */
const ACPI_EXCEPTION_INFO *
const char *
AcpiUtValidateException (
    ACPI_STATUS             Status);

BOOLEAN
AcpiUtIsPciRootBridge (
    char                    *Id);

BOOLEAN
AcpiUtIsAmlTable (
    ACPI_TABLE_HEADER       *Table);

ACPI_STATUS
AcpiUtAllocateOwnerId (
    ACPI_OWNER_ID           *OwnerId);

void
AcpiUtReleaseOwnerId (
    ACPI_OWNER_ID           *OwnerId);

ACPI_STATUS
AcpiUtWalkPackageTree (
    ACPI_OPERAND_OBJECT     *SourceObject,
    void                    *TargetObject,
    ACPI_PKG_CALLBACK       WalkCallback,
    void                    *Context);

void
AcpiUtStrupr (
    char                    *SrcString);

void
AcpiUtStrlwr (
    char                    *SrcString);

void
AcpiUtPrintString (
    char                    *String,
    UINT8                   MaxLength);

BOOLEAN
AcpiUtValidAcpiName (
    UINT32                  Name);

void
AcpiUtRepairName (
```



```
    char                    *Name);

BOOLEAN
AcpiUtValidAcpiChar (
    char                    Character,
    UINT32                  Position);

ACPI_STATUS
AcpiUtStrtoul64 (
    char                    *String,
    UINT32                  Base,
    UINT64                  *RetInteger);

/* Values for Base above (16=Hex, 10=Decimal) */

#define ACPI_ANY_BASE        0

UINT32
AcpiUtDwordByteSwap (
    UINT32                  Value);

void
AcpiUtSetIntegerWidth (
    UINT8                   Revision);

#ifdef ACPI_DEBUG_OUTPUT
void
AcpiUtDisplayInitPathname (
    UINT8                   Type,
    ACPI_NAMESPACE_NODE     *ObjHandle,
    char                    *Path);
#endif


/*
 * utownerid - Support for Table/Method Owner IDs
 */
ACPI_STATUS
AcpiUtAllocateOwnerId (
    ACPI_OWNER_ID           *OwnerId);

void
AcpiUtReleaseOwnerId (
    ACPI_OWNER_ID           *OwnerId);


/*
 * utresrc
 */
ACPI_STATUS
AcpiUtWalkAmlResources (
    ACPI_WALK_STATE         *WalkState,
    UINT8                   *Aml,
    ACPI_SIZE               AmlLength,
    ACPI_WALK_AML_CALLBACK  UserFunction,
    void                    **Context);
    void                    *Context);

ACPI_STATUS
AcpiUtValidateResource (
    ACPI_WALK_STATE         *WalkState,
    void                    *Aml,
    UINT8                   *ReturnIndex);

UINT32
AcpiUtGetDescriptorLength (
```

```
865     void                    *Aml);

867 UINT16
868 AcpiUtGetResourceLength (
869     void                    *Aml);

871 UINT8
872 AcpiUtGetResourceHeaderLength (
873     void                    *Aml);

875 UINT8
876 AcpiUtGetResourceType (
877     void                    *Aml);

879 ACPI_STATUS
880 AcpiUtGetResourceEndTag (
881     ACPI_OPERAND_OBJECT     *ObjDesc,
882     UINT8                   **EndTag);


885 /*
886  * utstring - String and character utilities
887  */
888 void
889 AcpiUtStrupr (
890     char                    *SrcString);

892 void
893 AcpiUtStrlwr (
894     char                    *SrcString);

896 int
897 AcpiUtStricmp (
898     char                    *String1,
899     char                    *String2);

901 ACPI_STATUS
902 AcpiUtStrtoul64 (
903     char                    *String,
904     UINT32                  Base,
905     UINT64                  *RetInteger);

907 void
908 AcpiUtPrintString (
909     char                    *String,
910     UINT16                  MaxLength);

912 void
913 UtConvertBackslashes (
914     char                    *Pathname);

916 BOOLEAN
917 AcpiUtValidAcpiName (
918     char                    *Name);

920 BOOLEAN
921 AcpiUtValidAcpiChar (
922     char                    Character,
923     UINT32                  Position);

925 void
926 AcpiUtRepairName (
927     char                    *Name);

929 #if defined (ACPI_DEBUGGER) || defined (ACPI_APPLICATION)
930 BOOLEAN
```



```
931 AcpiUtSafeStrcpy (
932     char                    *Dest,
933     ACPI_SIZE               DestSize,
934     char                    *Source);

936 BOOLEAN
937 AcpiUtSafeStrcat (
938     char                    *Dest,
939     ACPI_SIZE               DestSize,
940     char                    *Source);

942 BOOLEAN
943 AcpiUtSafeStrncat (
944     char                    *Dest,
945     ACPI_SIZE               DestSize,
946     char                    *Source,
947     ACPI_SIZE               MaxTransferLength);
948 #endif


951 /*
952  * utmutex - mutex support
953  */
954 ACPI_STATUS
955 AcpiUtMutexInitialize (
956     void);

958 void
959 AcpiUtMutexTerminate (
960     void);

962 ACPI_STATUS
963 AcpiUtAcquireMutex (
964     ACPI_MUTEX_HANDLE       MutexId);

966 ACPI_STATUS
967 AcpiUtReleaseMutex (
968     ACPI_MUTEX_HANDLE       MutexId);


971 /*
972  * utalloc - memory allocation and object caching
973  */
974 ACPI_STATUS
975 AcpiUtCreateCaches (
976     void);

978 ACPI_STATUS
979 AcpiUtDeleteCaches (
980     void);

982 ACPI_STATUS
983 AcpiUtValidateBuffer (
984     ACPI_BUFFER             *Buffer);

986 ACPI_STATUS
987 AcpiUtInitializeBuffer (
988     ACPI_BUFFER             *Buffer,
989     ACPI_SIZE               RequiredLength);

847 void *
848 AcpiUtAllocate (
849     ACPI_SIZE               Size,
850     UINT32                  Component,
851     const char              *Module,
852     UINT32                  Line);
```

```
 854 void *
 855 AcpiUtAllocateZeroed (
 856     ACPI_SIZE               Size,
 857     UINT32                  Component,
 858     const char              *Module,
 859     UINT32                  Line);
```

```
 991 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
 992 void *
 993 AcpiUtAllocateAndTrack (
 994     ACPI_SIZE               Size,
 995     UINT32                  Component,
 996     const char              *Module,
 997     UINT32                  Line);

 999 void *
1000 AcpiUtAllocateZeroedAndTrack (
1001     ACPI_SIZE               Size,
1002     UINT32                  Component,
1003     const char              *Module,
1004     UINT32                  Line);

1006 void
1007 AcpiUtFreeAndTrack (
1008     void                    *Address,
1009     UINT32                  Component,
1010     const char              *Module,
1011     UINT32                  Line);

1013 void
1014 AcpiUtDumpAllocationInfo (
1015     void);

1017 void
1018 AcpiUtDumpAllocations (
1019     UINT32                  Component,
1020     const char              *Module);

1022 ACPI_STATUS
1023 AcpiUtCreateList (
1024     char                    *ListName,
1025     UINT16                  ObjectSize,
1026     ACPI_MEMORY_LIST        **ReturnCache);

1028 #endif /* ACPI_DBG_TRACK_ALLOCATIONS */
```

```
1030 /*
1031  * utaddress - address range check
1032  */
1033 ACPI_STATUS
1034 AcpiUtAddAddressRange (
1035     ACPI_ADR_SPACE_TYPE     SpaceId,
1036     ACPI_PHYSICAL_ADDRESS   Address,
1037     UINT32                  Length,
1038     ACPI_NAMESPACE_NODE     *RegionNode);
```

```
1040 void
1041 AcpiUtRemoveAddressRange (
1042     ACPI_ADR_SPACE_TYPE     SpaceId,
1043     ACPI_NAMESPACE_NODE     *RegionNode);

1045 UINT32
1046 AcpiUtCheckAddressRange (
1047     ACPI_ADR_SPACE_TYPE     SpaceId,
1048     ACPI_PHYSICAL_ADDRESS   Address,
```



```
1049     UINT32                  Length,
1050     BOOLEAN                 Warn);

1052 void
1053 AcpiUtDeleteAddressLists (
1054     void);

1056 /*
1057  * utxferror - various error/warning output functions
1058  */
1059 void ACPI_INTERNAL_VAR_XFACE
1060 AcpiUtPredefinedWarning (
1061     const char              *ModuleName,
1062     UINT32                  LineNumber,
1063     char                    *Pathname,
1064     UINT8                   NodeFlags,
1065     const char              *Format,
1066     ...);

1068 void ACPI_INTERNAL_VAR_XFACE
1069 AcpiUtPredefinedInfo (
1070     const char              *ModuleName,
1071     UINT32                  LineNumber,
1072     char                    *Pathname,
1073     UINT8                   NodeFlags,
1074     const char              *Format,
1075     ...);

1077 void ACPI_INTERNAL_VAR_XFACE
1078 AcpiUtPredefinedBiosError (
1079     const char              *ModuleName,
1080     UINT32                  LineNumber,
1081     char                    *Pathname,
1082     UINT8                   NodeFlags,
1083     const char              *Format,
1084     ...);

1086 void
1087 AcpiUtNamespaceError (
1088     const char              *ModuleName,
1089     UINT32                  LineNumber,
1090     const char              *InternalName,
1091     ACPI_STATUS             LookupStatus);

1093 void
1094 AcpiUtMethodError (
1095     const char              *ModuleName,
1096     UINT32                  LineNumber,
1097     const char              *Message,
1098     ACPI_NAMESPACE_NODE     *Node,
1099     const char              *Path,
1100     ACPI_STATUS             LookupStatus);

1102 #endif /* _ACUTILS_H */
```

```
**********************************************************
   19946 Thu Dec 26 13:49:59 2013
new/usr/src/common/acpica/include/amlcode.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: amlcode.h - Definitions for AML, as included in "definition blocks"
   4  *                   Declarations and definitions contained herein are derived
   5  *                   directly from the ACPI specification.
   6  *
   7  *****************************************************************************/

   9 /*
  10  * Copyright (C) 2000 - 2013, Intel Corp.
  10  * Copyright (C) 2000 - 2011, Intel Corp.
  11  * All rights reserved.
  12  *
  13  * Redistribution and use in source and binary forms, with or without
  14  * modification, are permitted provided that the following conditions
  15  * are met:
  16  * 1. Redistributions of source code must retain the above copyright
  17  *    notice, this list of conditions, and the following disclaimer,
  18  *    without modification.
  19  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  20  *    substantially similar to the "NO WARRANTY" disclaimer below
  21  *    ("Disclaimer") and any redistribution must be conditioned upon
  22  *    including a substantially similar Disclaimer requirement for further
  23  *    binary redistribution.
  24  * 3. Neither the names of the above-listed copyright holders nor the names
  25  *    of any contributors may be used to endorse or promote products derived
  26  *    from this software without specific prior written permission.
  27  *
  28  * Alternatively, this software may be distributed under the terms of the
  29  * GNU General Public License ("GPL") version 2 as published by the Free
  30  * Software Foundation.
  31  *
  32  * NO WARRANTY
  33  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  34  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  35  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  36  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  37  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  38  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  39  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  40  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  41  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  42  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  43  * POSSIBILITY OF SUCH DAMAGES.
  44  */

  46 #ifndef __AMLCODE_H__
  47 #define __AMLCODE_H__

  49 /* primary opcodes */

  51 #define AML_NULL_CHAR               (UINT16) 0x00

  53 #define AML_ZERO_OP                 (UINT16) 0x00
  54 #define AML_ONE_OP                  (UINT16) 0x01
  55 #define AML_UNASSIGNED              (UINT16) 0x02
  56 #define AML_ALIAS_OP                (UINT16) 0x06
  57 #define AML_NAME_OP                 (UINT16) 0x08
  58 #define AML_BYTE_OP                 (UINT16) 0x0a
  59 #define AML_WORD_OP                 (UINT16) 0x0b
```



```
  60 #define AML_DWORD_OP                (UINT16) 0x0c
  61 #define AML_STRING_OP               (UINT16) 0x0d
  62 #define AML_QWORD_OP                (UINT16) 0x0e     /* ACPI 2.0 */
  63 #define AML_SCOPE_OP                (UINT16) 0x10
  64 #define AML_BUFFER_OP               (UINT16) 0x11
  65 #define AML_PACKAGE_OP              (UINT16) 0x12
  66 #define AML_VAR_PACKAGE_OP          (UINT16) 0x13     /* ACPI 2.0 */
  67 #define AML_METHOD_OP               (UINT16) 0x14
  68 #define AML_DUAL_NAME_PREFIX        (UINT16) 0x2e
  69 #define AML_MULTI_NAME_PREFIX_OP    (UINT16) 0x2f
  70 #define AML_NAME_CHAR_SUBSEQ        (UINT16) 0x30
  71 #define AML_NAME_CHAR_FIRST         (UINT16) 0x41
  72 #define AML_EXTENDED_OP_PREFIX      (UINT16) 0x5b
  73 #define AML_ROOT_PREFIX             (UINT16) 0x5c
  74 #define AML_PARENT_PREFIX           (UINT16) 0x5e
  75 #define AML_LOCAL_OP                (UINT16) 0x60
  76 #define AML_LOCAL0                  (UINT16) 0x60
  77 #define AML_LOCAL1                  (UINT16) 0x61
  78 #define AML_LOCAL2                  (UINT16) 0x62
  79 #define AML_LOCAL3                  (UINT16) 0x63
  80 #define AML_LOCAL4                  (UINT16) 0x64
  81 #define AML_LOCAL5                  (UINT16) 0x65
  82 #define AML_LOCAL6                  (UINT16) 0x66
  83 #define AML_LOCAL7                  (UINT16) 0x67
  84 #define AML_ARG_OP                  (UINT16) 0x68
  85 #define AML_ARG0                    (UINT16) 0x68
  86 #define AML_ARG1                    (UINT16) 0x69
  87 #define AML_ARG2                    (UINT16) 0x6a
  88 #define AML_ARG3                    (UINT16) 0x6b
  89 #define AML_ARG4                    (UINT16) 0x6c
  90 #define AML_ARG5                    (UINT16) 0x6d
  91 #define AML_ARG6                    (UINT16) 0x6e
  92 #define AML_STORE_OP                (UINT16) 0x70
  93 #define AML_REF_OF_OP               (UINT16) 0x71
  94 #define AML_ADD_OP                  (UINT16) 0x72
  95 #define AML_CONCAT_OP               (UINT16) 0x73
  96 #define AML_SUBTRACT_OP             (UINT16) 0x74
  97 #define AML_INCREMENT_OP            (UINT16) 0x75
  98 #define AML_DECREMENT_OP            (UINT16) 0x76
  99 #define AML_MULTIPLY_OP             (UINT16) 0x77
 100 #define AML_DIVIDE_OP               (UINT16) 0x78
 101 #define AML_SHIFT_LEFT_OP           (UINT16) 0x79
 102 #define AML_SHIFT_RIGHT_OP          (UINT16) 0x7a
 103 #define AML_BIT_AND_OP              (UINT16) 0x7b
 104 #define AML_BIT_NAND_OP             (UINT16) 0x7c
 105 #define AML_BIT_OR_OP               (UINT16) 0x7d
 106 #define AML_BIT_NOR_OP              (UINT16) 0x7e
 107 #define AML_BIT_XOR_OP              (UINT16) 0x7f
 108 #define AML_BIT_NOT_OP              (UINT16) 0x80
 109 #define AML_FIND_SET_LEFT_BIT_OP    (UINT16) 0x81
 110 #define AML_FIND_SET_RIGHT_BIT_OP   (UINT16) 0x82
 111 #define AML_DEREF_OF_OP             (UINT16) 0x83
 112 #define AML_CONCAT_RES_OP           (UINT16) 0x84     /* ACPI 2.0 */
 113 #define AML_MOD_OP                  (UINT16) 0x85     /* ACPI 2.0 */
 114 #define AML_NOTIFY_OP               (UINT16) 0x86
 115 #define AML_SIZE_OF_OP              (UINT16) 0x87
 116 #define AML_INDEX_OP                (UINT16) 0x88
 117 #define AML_MATCH_OP                (UINT16) 0x89
 118 #define AML_CREATE_DWORD_FIELD_OP   (UINT16) 0x8a
 119 #define AML_CREATE_WORD_FIELD_OP    (UINT16) 0x8b
 120 #define AML_CREATE_BYTE_FIELD_OP    (UINT16) 0x8c
 121 #define AML_CREATE_BIT_FIELD_OP     (UINT16) 0x8d
 122 #define AML_TYPE_OP                 (UINT16) 0x8e
 123 #define AML_CREATE_QWORD_FIELD_OP   (UINT16) 0x8f     /* ACPI 2.0 */
 124 #define AML_LAND_OP                 (UINT16) 0x90
 125 #define AML_LOR_OP                  (UINT16) 0x91
```

```
126 #define AML_LNOT_OP                 (UINT16) 0x92
127 #define AML_LEQUAL_OP               (UINT16) 0x93
128 #define AML_LGREATER_OP             (UINT16) 0x94
129 #define AML_LLESS_OP                (UINT16) 0x95
130 #define AML_TO_BUFFER_OP            (UINT16) 0x96     /* ACPI 2.0 */
131 #define AML_TO_DECSTRING_OP         (UINT16) 0x97     /* ACPI 2.0 */
132 #define AML_TO_HEXSTRING_OP         (UINT16) 0x98     /* ACPI 2.0 */
133 #define AML_TO_INTEGER_OP           (UINT16) 0x99     /* ACPI 2.0 */
134 #define AML_TO_STRING_OP            (UINT16) 0x9c     /* ACPI 2.0 */
135 #define AML_COPY_OP                 (UINT16) 0x9d     /* ACPI 2.0 */
136 #define AML_MID_OP                  (UINT16) 0x9e     /* ACPI 2.0 */
137 #define AML_CONTINUE_OP             (UINT16) 0x9f     /* ACPI 2.0 */
138 #define AML_IF_OP                   (UINT16) 0xa0
139 #define AML_ELSE_OP                 (UINT16) 0xa1
140 #define AML_WHILE_OP                (UINT16) 0xa2
141 #define AML_NOOP_OP                 (UINT16) 0xa3
142 #define AML_RETURN_OP               (UINT16) 0xa4
143 #define AML_BREAK_OP                (UINT16) 0xa5
144 #define AML_BREAK_POINT_OP          (UINT16) 0xcc
145 #define AML_ONES_OP                 (UINT16) 0xff

147 /* prefixed opcodes */

149 #define AML_EXTENDED_OPCODE         (UINT16) 0x5b00     /* prefix for 2-byte opc

151 #define AML_MUTEX_OP                (UINT16) 0x5b01
152 #define AML_EVENT_OP                (UINT16) 0x5b02
153 #define AML_SHIFT_RIGHT_BIT_OP      (UINT16) 0x5b10
154 #define AML_SHIFT_LEFT_BIT_OP       (UINT16) 0x5b11
155 #define AML_COND_REF_OF_OP          (UINT16) 0x5b12
156 #define AML_CREATE_FIELD_OP         (UINT16) 0x5b13
157 #define AML_LOAD_TABLE_OP           (UINT16) 0x5b1f     /* ACPI 2.0 */
158 #define AML_LOAD_OP                 (UINT16) 0x5b20
159 #define AML_STALL_OP                (UINT16) 0x5b21
160 #define AML_SLEEP_OP                (UINT16) 0x5b22
161 #define AML_ACQUIRE_OP              (UINT16) 0x5b23
162 #define AML_SIGNAL_OP               (UINT16) 0x5b24
163 #define AML_WAIT_OP                 (UINT16) 0x5b25
164 #define AML_RESET_OP                (UINT16) 0x5b26
165 #define AML_RELEASE_OP              (UINT16) 0x5b27
166 #define AML_FROM_BCD_OP             (UINT16) 0x5b28
167 #define AML_TO_BCD_OP               (UINT16) 0x5b29
168 #define AML_UNLOAD_OP               (UINT16) 0x5b2a
169 #define AML_REVISION_OP             (UINT16) 0x5b30
170 #define AML_DEBUG_OP                (UINT16) 0x5b31
171 #define AML_FATAL_OP                (UINT16) 0x5b32
172 #define AML_TIMER_OP                (UINT16) 0x5b33     /* ACPI 3.0 */
173 #define AML_REGION_OP               (UINT16) 0x5b80
174 #define AML_FIELD_OP                (UINT16) 0x5b81
175 #define AML_DEVICE_OP               (UINT16) 0x5b82
176 #define AML_PROCESSOR_OP            (UINT16) 0x5b83
177 #define AML_POWER_RES_OP            (UINT16) 0x5b84
178 #define AML_THERMAL_ZONE_OP         (UINT16) 0x5b85
179 #define AML_INDEX_FIELD_OP          (UINT16) 0x5b86
180 #define AML_BANK_FIELD_OP           (UINT16) 0x5b87
181 #define AML_DATA_REGION_OP          (UINT16) 0x5b88     /* ACPI 2.0 */


184 /*
185  * Combination opcodes (actually two one-byte opcodes)
186  * Used by the disassembler and iASL compiler
187  */
188 #define AML_LGREATEREQUAL_OP        (UINT16) 0x9295
189 #define AML_LLESSEQUAL_OP           (UINT16) 0x9294
190 #define AML_LNOTEQUAL_OP            (UINT16) 0x9293
```



```
193 /*
194  * Opcodes for "Field" operators
195  */
196 #define AML_FIELD_OFFSET_OP         (UINT8) 0x00
197 #define AML_FIELD_ACCESS_OP         (UINT8) 0x01
198 #define AML_FIELD_CONNECTION_OP     (UINT8) 0x02        /* ACPI 5.0 */
199 #define AML_FIELD_EXT_ACCESS_OP     (UINT8) 0x03        /* ACPI 5.0 */


202 /*
203  * Internal opcodes
204  * Use only "Unknown" AML opcodes, don't attempt to use
205  * any valid ACPI ASCII values (A-Z, 0-9, '-')
206  */
207 #define AML_INT_NAMEPATH_OP         (UINT16) 0x002d
208 #define AML_INT_NAMEDFIELD_OP       (UINT16) 0x0030
209 #define AML_INT_RESERVEDFIELD_OP    (UINT16) 0x0031
210 #define AML_INT_ACCESSFIELD_OP      (UINT16) 0x0032
211 #define AML_INT_BYTELIST_OP         (UINT16) 0x0033
212 #define AML_INT_STATICSTRING_OP     (UINT16) 0x0034
213 #define AML_INT_METHODCALL_OP       (UINT16) 0x0035
214 #define AML_INT_RETURN_VALUE_OP     (UINT16) 0x0036
215 #define AML_INT_EVAL_SUBTREE_OP     (UINT16) 0x0037
216 #define AML_INT_CONNECTION_OP       (UINT16) 0x0038
217 #define AML_INT_EXTACCESSFIELD_OP   (UINT16) 0x0039


219 #define ARG_NONE                    0x0

221 /*
222  * Argument types for the AML Parser
223  * Each field in the ArgTypes UINT32 is 5 bits, allowing for a maximum of 6 argu
224  * There can be up to 31 unique argument types
225  * Zero is reserved as end-of-list indicator
226  */
227 #define ARGP_BYTEDATA               0x01
228 #define ARGP_BYTELIST               0x02
229 #define ARGP_CHARLIST               0x03
230 #define ARGP_DATAOBJ                0x04
231 #define ARGP_DATAOBJLIST            0x05
232 #define ARGP_DWORDDATA              0x06
233 #define ARGP_FIELDLIST              0x07
234 #define ARGP_NAME                   0x08
235 #define ARGP_NAMESTRING             0x09
236 #define ARGP_OBJLIST                0x0A
237 #define ARGP_PKGLENGTH              0x0B
238 #define ARGP_SUPERNAME              0x0C
239 #define ARGP_TARGET                 0x0D
240 #define ARGP_TERMARG                0x0E
241 #define ARGP_TERMLIST               0x0F
242 #define ARGP_WORDDATA               0x10
243 #define ARGP_QWORDDATA              0x11
244 #define ARGP_SIMPLENAME             0x12

246 /*
247  * Resolved argument types for the AML Interpreter
248  * Each field in the ArgTypes UINT32 is 5 bits, allowing for a maximum of 6 argu
249  * There can be up to 31 unique argument types (0 is end-of-arg-list indicator)
250  *
251  * Note1: These values are completely independent from the ACPI_TYPEs
252  *        i.e., ARGI_INTEGER != ACPI_TYPE_INTEGER
253  *
254  * Note2: If and when 5 bits becomes insufficient, it would probably be best
255  * to convert to a 6-byte array of argument types, allowing 8 bits per argument.
256  */
```

```
258 /* Single, simple types */

260 #define ARGI_ANYTYPE                0x01    /* Don't care */
261 #define ARGI_PACKAGE                0x02
262 #define ARGI_EVENT                  0x03
263 #define ARGI_MUTEX                  0x04
264 #define ARGI_DDBHANDLE              0x05

266 /* Interchangeable types (via implicit conversion) */

268 #define ARGI_INTEGER                0x06
269 #define ARGI_STRING                 0x07
270 #define ARGI_BUFFER                 0x08
271 #define ARGI_BUFFER_OR_STRING       0x09    /* Used by MID op only */
272 #define ARGI_COMPUTEDATA            0x0A    /* Buffer, String, or Integer */

274 /* Reference objects */

276 #define ARGI_INTEGER_REF            0x0B
277 #define ARGI_OBJECT_REF             0x0C
278 #define ARGI_DEVICE_REF             0x0D
279 #define ARGI_REFERENCE              0x0E
280 #define ARGI_TARGETREF              0x0F    /* Target, subject to implicit conve
281 #define ARGI_FIXED_TARGET           0x10    /* Target, no implicit conversion */
282 #define ARGI_SIMPLE_TARGET          0x11    /* Name, Local, Arg -- no implicit c

284 /* Multiple/complex types */

286 #define ARGI_DATAOBJECT             0x12    /* Buffer, String, package or refere
287 #define ARGI_COMPLEXOBJ             0x13    /* Buffer, String, or package (Used
288 #define ARGI_REF_OR_STRING          0x14    /* Reference or String (Used by DERE
289 #define ARGI_REGION_OR_BUFFER       0x15    /* Used by LOAD op only */
290 #define ARGI_DATAREFOBJ             0x16

292 /* Note: types above can expand to 0x1F maximum */

294 #define ARGI_INVALID_OPCODE         0xFFFFFFFF


297 /*
298  * hash offsets
299  */
300 #define AML_EXTOP_HASH_OFFSET       22
301 #define AML_LNOT_HASH_OFFSET        19


304 /*
305  * opcode groups and types
306  */
307 #define OPGRP_NAMED                 0x01
308 #define OPGRP_FIELD                 0x02
309 #define OPGRP_BYTELIST              0x04


312 /*
313  * Opcode information
314  */

316 /* Opcode flags */

318 #define AML_LOGICAL                 0x0001
319 #define AML_LOGICAL_NUMERIC         0x0002
320 #define AML_MATH                    0x0004
321 #define AML_CREATE                  0x0008
322 #define AML_FIELD                   0x0010
```



```
323 #define AML_DEFER                   0x0020
324 #define AML_NAMED                   0x0040
325 #define AML_NSNODE                  0x0080
326 #define AML_NSOPCODE                0x0100
327 #define AML_NSOBJECT                0x0200
328 #define AML_HAS_RETVAL              0x0400
329 #define AML_HAS_TARGET              0x0800
330 #define AML_HAS_ARGS                0x1000
331 #define AML_CONSTANT                0x2000
332 #define AML_NO_OPERAND_RESOLVE      0x4000

334 /* Convenient flag groupings */

336 #define AML_FLAGS_EXEC_0A_0T_1R                                     AML_HAS_RETV
337 #define AML_FLAGS_EXEC_1A_0T_0R     AML_HAS_ARGS
338 #define AML_FLAGS_EXEC_1A_0T_1R     AML_HAS_ARGS |                  AML_HAS_RETV
339 #define AML_FLAGS_EXEC_1A_1T_0R     AML_HAS_ARGS | AML_HAS_TARGET
340 #define AML_FLAGS_EXEC_1A_1T_1R     AML_HAS_ARGS | AML_HAS_TARGET | AML_HAS_RETV
341 #define AML_FLAGS_EXEC_2A_0T_0R     AML_HAS_ARGS
342 #define AML_FLAGS_EXEC_2A_0T_1R     AML_HAS_ARGS |                  AML_HAS_RETV
343 #define AML_FLAGS_EXEC_2A_1T_1R     AML_HAS_ARGS | AML_HAS_TARGET | AML_HAS_RETV
344 #define AML_FLAGS_EXEC_2A_2T_1R     AML_HAS_ARGS | AML_HAS_TARGET | AML_HAS_RETV
345 #define AML_FLAGS_EXEC_3A_0T_0R     AML_HAS_ARGS
346 #define AML_FLAGS_EXEC_3A_1T_1R     AML_HAS_ARGS | AML_HAS_TARGET | AML_HAS_RETV
347 #define AML_FLAGS_EXEC_6A_0T_1R     AML_HAS_ARGS |                  AML_HAS_RETV


350 /*
351  * The opcode Type is used in a dispatch table, do not change
352  * without updating the table.
353  */
354 #define AML_TYPE_EXEC_0A_0T_1R      0x00
355 #define AML_TYPE_EXEC_1A_0T_0R      0x01 /* Monadic1  */
356 #define AML_TYPE_EXEC_1A_0T_1R      0x02 /* Monadic2  */
357 #define AML_TYPE_EXEC_1A_1T_0R      0x03
358 #define AML_TYPE_EXEC_1A_1T_1R      0x04 /* Monadic2R */
359 #define AML_TYPE_EXEC_2A_0T_0R      0x05 /* Dyadic1   */
360 #define AML_TYPE_EXEC_2A_0T_1R      0x06 /* Dyadic2   */
361 #define AML_TYPE_EXEC_2A_1T_1R      0x07 /* Dyadic2R  */
362 #define AML_TYPE_EXEC_2A_2T_1R      0x08
363 #define AML_TYPE_EXEC_3A_0T_0R      0x09
364 #define AML_TYPE_EXEC_3A_1T_1R      0x0A
365 #define AML_TYPE_EXEC_6A_0T_1R      0x0B
366 /* End of types used in dispatch table */

368 #define AML_TYPE_LITERAL            0x0B
369 #define AML_TYPE_CONSTANT           0x0C
370 #define AML_TYPE_METHOD_ARGUMENT    0x0D
371 #define AML_TYPE_LOCAL_VARIABLE     0x0E
372 #define AML_TYPE_DATA_TERM          0x0F

374 /* Generic for an op that returns a value */

376 #define AML_TYPE_METHOD_CALL        0x10

378 /* Misc */

380 #define AML_TYPE_CREATE_FIELD       0x11
381 #define AML_TYPE_CREATE_OBJECT      0x12
382 #define AML_TYPE_CONTROL            0x13
383 #define AML_TYPE_NAMED_NO_OBJ       0x14
384 #define AML_TYPE_NAMED_FIELD        0x15
385 #define AML_TYPE_NAMED_SIMPLE       0x16
386 #define AML_TYPE_NAMED_COMPLEX      0x17
387 #define AML_TYPE_RETURN             0x18
```

```
 389 #define AML_TYPE_UNDEFINED          0x19
 390 #define AML_TYPE_BOGUS              0x1A

 392 /* AML Package Length encodings */

 394 #define ACPI_AML_PACKAGE_TYPE1      0x40
 395 #define ACPI_AML_PACKAGE_TYPE2      0x4000
 396 #define ACPI_AML_PACKAGE_TYPE3      0x400000
 397 #define ACPI_AML_PACKAGE_TYPE4      0x40000000

 399 /*
 400  * Opcode classes
 401  */
 402 #define AML_CLASS_EXECUTE           0x00
 403 #define AML_CLASS_CREATE            0x01
 404 #define AML_CLASS_ARGUMENT          0x02
 405 #define AML_CLASS_NAMED_OBJECT      0x03
 406 #define AML_CLASS_CONTROL           0x04
 407 #define AML_CLASS_ASCII             0x05
 408 #define AML_CLASS_PREFIX            0x06
 409 #define AML_CLASS_INTERNAL          0x07
 410 #define AML_CLASS_RETURN_VALUE      0x08
 411 #define AML_CLASS_METHOD_CALL       0x09
 412 #define AML_CLASS_UNKNOWN           0x0A


 415 /* Comparison operation codes for MatchOp operator */

 417 typedef enum
 418 {
 419     MATCH_MTR                       = 0,
 420     MATCH_MEQ                       = 1,
 421     MATCH_MLE                       = 2,
 422     MATCH_MLT                       = 3,
 423     MATCH_MGE                       = 4,
 424     MATCH_MGT                       = 5

 426 } AML_MATCH_OPERATOR;
______unchanged_portion_omitted_


 484 /*
 485  * Field Access Attributes.
 486  * This byte is extracted from the AML via the
 487  * AccessAs keyword
 488  */
 489 typedef enum
 490 {
 491     AML_FIELD_ATTRIB_QUICK          = 0x02,
 492     AML_FIELD_ATTRIB_SEND_RCV       = 0x04,
 493     AML_FIELD_ATTRIB_BYTE           = 0x06,
 494     AML_FIELD_ATTRIB_WORD           = 0x08,
 495     AML_FIELD_ATTRIB_BLOCK          = 0x0A,
 496     AML_FIELD_ATTRIB_MULTIBYTE      = 0x0B,
 497     AML_FIELD_ATTRIB_WORD_CALL      = 0x0C,
 498     AML_FIELD_ATTRIB_BLOCK_CALL     = 0x0D,
 499     AML_FIELD_ATTRIB_RAW_BYTES      = 0x0E,
 500     AML_FIELD_ATTRIB_RAW_PROCESS    = 0x0F
 481     AML_FIELD_ATTRIB_SMB_QUICK      = 0x02,
 482     AML_FIELD_ATTRIB_SMB_SEND_RCV   = 0x04,
 483     AML_FIELD_ATTRIB_SMB_BYTE       = 0x06,
 484     AML_FIELD_ATTRIB_SMB_WORD       = 0x08,
 485     AML_FIELD_ATTRIB_SMB_BLOCK      = 0x0A,
 486     AML_FIELD_ATTRIB_SMB_WORD_CALL  = 0x0C,
 487     AML_FIELD_ATTRIB_SMB_BLOCK_CALL = 0x0D
```



```
 502 } AML_ACCESS_ATTRIBUTE;
______unchanged_portion_omitted_
```

```
**********************************************************
   19141 Thu Dec 26 13:49:59 2013
new/usr/src/common/acpica/include/amlresrc.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: amlresrc.h - AML resource descriptors
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

/* acpisrc:StructDefs -- for acpisrc conversion */

#ifndef __AMLRESRC_H
#define __AMLRESRC_H


/*
 * Resource descriptor tags, as defined in the ACPI specification.
 * Used to symbolically reference fields within a descriptor.
 */
#define ACPI_RESTAG_ADDRESS                 "_ADR"
#define ACPI_RESTAG_ALIGNMENT               "_ALN"
#define ACPI_RESTAG_ADDRESSSPACE            "_ASI"
#define ACPI_RESTAG_ACCESSSIZE              "_ASZ"
#define ACPI_RESTAG_TYPESPECIFICATTRIBUTES  "_ATT"
#define ACPI_RESTAG_BASEADDRESS             "_BAS"
#define ACPI_RESTAG_BUSMASTER               "_BM_"  /* Master(1), Slave(0) *
```



```
#define ACPI_RESTAG_DEBOUNCETIME            "_DBT"
#define ACPI_RESTAG_DECODE                  "_DEC"
#define ACPI_RESTAG_DEVICEPOLARITY          "_DPL"
#define ACPI_RESTAG_DMA                     "_DMA"
#define ACPI_RESTAG_DMATYPE                 "_TYP"  /* Compatible(0), A(1),
#define ACPI_RESTAG_DRIVESTRENGTH           "_DRS"
#define ACPI_RESTAG_ENDIANNESS              "_END"
#define ACPI_RESTAG_FLOWCONTROL             "_FLC"
#define ACPI_RESTAG_GRANULARITY             "_GRA"
#define ACPI_RESTAG_INTERRUPT               "_INT"
#define ACPI_RESTAG_INTERRUPTLEVEL          "_LL_"  /* ActiveLo(1), ActiveHi
#define ACPI_RESTAG_INTERRUPTSHARE          "_SHR"  /* Shareable(1), NoShare
#define ACPI_RESTAG_INTERRUPTTYPE           "_HE_"  /* Edge(1), Level(0) */
#define ACPI_RESTAG_IORESTRICTION           "_IOR"
#define ACPI_RESTAG_LENGTH                  "_LEN"
#define ACPI_RESTAG_LINE                    "_LIN"
#define ACPI_RESTAG_MEMATTRIBUTES           "_MTP"  /* Memory(0), Reserved(1
#define ACPI_RESTAG_MEMTYPE                 "_MEM"  /* NonCache(0), Cacheabl
#define ACPI_RESTAG_MAXADDR                 "_MAX"
#define ACPI_RESTAG_MINADDR                 "_MIN"
#define ACPI_RESTAG_MAXTYPE                 "_MAF"
#define ACPI_RESTAG_MINTYPE                 "_MIF"
#define ACPI_RESTAG_MODE                    "_MOD"
#define ACPI_RESTAG_PARITY                  "_PAR"
#define ACPI_RESTAG_PHASE                   "_PHA"
#define ACPI_RESTAG_PIN                     "_PIN"
#define ACPI_RESTAG_PINCONFIG               "_PPI"
#define ACPI_RESTAG_POLARITY                "_POL"
#define ACPI_RESTAG_REGISTERBITOFFSET       "_RBO"
#define ACPI_RESTAG_REGISTERBITWIDTH        "_RBW"
#define ACPI_RESTAG_RANGETYPE               "_RNG"
#define ACPI_RESTAG_READWRITETYPE           "_RW_"  /* ReadOnly(0), Writeabl
#define ACPI_RESTAG_LENGTH_RX               "_RXL"
#define ACPI_RESTAG_LENGTH_TX               "_TXL"
#define ACPI_RESTAG_SLAVEMODE               "_SLV"
#define ACPI_RESTAG_SPEED                   "_SPE"
#define ACPI_RESTAG_STOPBITS                "_STB"
#define ACPI_RESTAG_TRANSLATION             "_TRA"
#define ACPI_RESTAG_TRANSTYPE               "_TRS"  /* Sparse(1), Dense(0) *
#define ACPI_RESTAG_TYPE                    "_TTP"  /* Translation(1), Stati
#define ACPI_RESTAG_XFERTYPE                "_SIZ"  /* 8(0), 8And16(1), 16(2
#define ACPI_RESTAG_VENDORDATA              "_VEN"


/* Default sizes for "small" resource descriptors */

#define ASL_RDESC_IRQ_SIZE                  0x02
#define ASL_RDESC_DMA_SIZE                  0x02
#define ASL_RDESC_ST_DEPEND_SIZE            0x00
#define ASL_RDESC_END_DEPEND_SIZE           0x00
#define ASL_RDESC_IO_SIZE                   0x07
#define ASL_RDESC_FIXED_IO_SIZE             0x03
#define ASL_RDESC_FIXED_DMA_SIZE            0x05
#define ASL_RDESC_END_TAG_SIZE              0x01


typedef struct asl_resource_node
{
    UINT32                      BufferLength;
    void                        *Buffer;
    struct asl_resource_node    *Next;

} ASL_RESOURCE_NODE;


/* Macros used to generate AML resource length fields */
```

```
128 #define ACPI_AML_SIZE_LARGE(r)      (sizeof (r) - sizeof (AML_RESOURCE_LARGE_HEA
129 #define ACPI_AML_SIZE_SMALL(r)      (sizeof (r) - sizeof (AML_RESOURCE_SMALL_HEA

131 /*
132  * Resource descriptors defined in the ACPI specification.
133  *
134  * Packing/alignment must be BYTE because these descriptors
135  * are used to overlay the raw AML byte stream.
136  */
137 #pragma pack(1)

139 /*
140  * SMALL descriptors
141  */
142 #define AML_RESOURCE_SMALL_HEADER_COMMON \
143     UINT8                           DescriptorType;

145 typedef struct aml_resource_small_header
146 {
147     AML_RESOURCE_SMALL_HEADER_COMMON

149 } AML_RESOURCE_SMALL_HEADER;


152 typedef struct aml_resource_irq
153 {
154     AML_RESOURCE_SMALL_HEADER_COMMON
155     UINT16                          IrqMask;
156     UINT8                           Flags;

158 } AML_RESOURCE_IRQ;


161 typedef struct aml_resource_irq_noflags
162 {
163     AML_RESOURCE_SMALL_HEADER_COMMON
164     UINT16                          IrqMask;

166 } AML_RESOURCE_IRQ_NOFLAGS;


169 typedef struct aml_resource_dma
170 {
171     AML_RESOURCE_SMALL_HEADER_COMMON
172     UINT8                           DmaChannelMask;
173     UINT8                           Flags;

175 } AML_RESOURCE_DMA;


178 typedef struct aml_resource_start_dependent
179 {
180     AML_RESOURCE_SMALL_HEADER_COMMON
181     UINT8                           Flags;

183 } AML_RESOURCE_START_DEPENDENT;


186 typedef struct aml_resource_start_dependent_noprio
187 {
188     AML_RESOURCE_SMALL_HEADER_COMMON

190 } AML_RESOURCE_START_DEPENDENT_NOPRIO;
```



```
193 typedef struct aml_resource_end_dependent
194 {
195     AML_RESOURCE_SMALL_HEADER_COMMON

197 } AML_RESOURCE_END_DEPENDENT;


200 typedef struct aml_resource_io
201 {
202     AML_RESOURCE_SMALL_HEADER_COMMON
203     UINT8                           Flags;
204     UINT16                          Minimum;
205     UINT16                          Maximum;
206     UINT8                           Alignment;
207     UINT8                           AddressLength;

209 } AML_RESOURCE_IO;


212 typedef struct aml_resource_fixed_io
213 {
214     AML_RESOURCE_SMALL_HEADER_COMMON
215     UINT16                          Address;
216     UINT8                           AddressLength;

218 } AML_RESOURCE_FIXED_IO;


221 typedef struct aml_resource_vendor_small
222 {
223     AML_RESOURCE_SMALL_HEADER_COMMON

225 } AML_RESOURCE_VENDOR_SMALL;


228 typedef struct aml_resource_end_tag
229 {
230     AML_RESOURCE_SMALL_HEADER_COMMON
231     UINT8                           Checksum;

233 } AML_RESOURCE_END_TAG;


236 typedef struct aml_resource_fixed_dma
237 {
238     AML_RESOURCE_SMALL_HEADER_COMMON
239     UINT16                          RequestLines;
240     UINT16                          Channels;
241     UINT8                           Width;

243 } AML_RESOURCE_FIXED_DMA;


246 /*
247  * LARGE descriptors
248  */
249 #define AML_RESOURCE_LARGE_HEADER_COMMON \
250     UINT8                           DescriptorType;\
251     UINT16                          ResourceLength;

253 typedef struct aml_resource_large_header
254 {
255     AML_RESOURCE_LARGE_HEADER_COMMON

257 } AML_RESOURCE_LARGE_HEADER;
```

```
260 /* General Flags for address space resource descriptors */

262 #define ACPI_RESOURCE_FLAG_DEC      2
263 #define ACPI_RESOURCE_FLAG_MIF      4
264 #define ACPI_RESOURCE_FLAG_MAF      8

266 typedef struct aml_resource_memory24
267 {
268     AML_RESOURCE_LARGE_HEADER_COMMON
269     UINT8                           Flags;
270     UINT16                          Minimum;
271     UINT16                          Maximum;
272     UINT16                          Alignment;
273     UINT16                          AddressLength;

275 } AML_RESOURCE_MEMORY24;


278 typedef struct aml_resource_vendor_large
279 {
280     AML_RESOURCE_LARGE_HEADER_COMMON

282 } AML_RESOURCE_VENDOR_LARGE;


285 typedef struct aml_resource_memory32
286 {
287     AML_RESOURCE_LARGE_HEADER_COMMON
288     UINT8                           Flags;
289     UINT32                          Minimum;
290     UINT32                          Maximum;
291     UINT32                          Alignment;
292     UINT32                          AddressLength;

294 } AML_RESOURCE_MEMORY32;


297 typedef struct aml_resource_fixed_memory32
298 {
299     AML_RESOURCE_LARGE_HEADER_COMMON
300     UINT8                           Flags;
301     UINT32                          Address;
302     UINT32                          AddressLength;

304 } AML_RESOURCE_FIXED_MEMORY32;


307 #define AML_RESOURCE_ADDRESS_COMMON \
308     UINT8                           ResourceType; \
309     UINT8                           Flags; \
310     UINT8                           SpecificFlags;


313 typedef struct aml_resource_address
314 {
315     AML_RESOURCE_LARGE_HEADER_COMMON
316     AML_RESOURCE_ADDRESS_COMMON

318 } AML_RESOURCE_ADDRESS;


321 typedef struct aml_resource_extended_address64
322 {
323     AML_RESOURCE_LARGE_HEADER_COMMON
324     AML_RESOURCE_ADDRESS_COMMON
```



```
325     UINT8                           RevisionID;
326     UINT8                           Reserved;
327     UINT64                          Granularity;
328     UINT64                          Minimum;
329     UINT64                          Maximum;
330     UINT64                          TranslationOffset;
331     UINT64                          AddressLength;
332     UINT64                          TypeSpecific;

334 } AML_RESOURCE_EXTENDED_ADDRESS64;

336 #define AML_RESOURCE_EXTENDED_ADDRESS_REVISION          1       /* ACPI 3.0 */


339 typedef struct aml_resource_address64
340 {
341     AML_RESOURCE_LARGE_HEADER_COMMON
342     AML_RESOURCE_ADDRESS_COMMON
343     UINT64                          Granularity;
344     UINT64                          Minimum;
345     UINT64                          Maximum;
346     UINT64                          TranslationOffset;
347     UINT64                          AddressLength;

349 } AML_RESOURCE_ADDRESS64;


352 typedef struct aml_resource_address32
353 {
354     AML_RESOURCE_LARGE_HEADER_COMMON
355     AML_RESOURCE_ADDRESS_COMMON
356     UINT32                          Granularity;
357     UINT32                          Minimum;
358     UINT32                          Maximum;
359     UINT32                          TranslationOffset;
360     UINT32                          AddressLength;

362 } AML_RESOURCE_ADDRESS32;


365 typedef struct aml_resource_address16
366 {
367     AML_RESOURCE_LARGE_HEADER_COMMON
368     AML_RESOURCE_ADDRESS_COMMON
369     UINT16                          Granularity;
370     UINT16                          Minimum;
371     UINT16                          Maximum;
372     UINT16                          TranslationOffset;
373     UINT16                          AddressLength;

375 } AML_RESOURCE_ADDRESS16;


378 typedef struct aml_resource_extended_irq
379 {
380     AML_RESOURCE_LARGE_HEADER_COMMON
381     UINT8                           Flags;
382     UINT8                           InterruptCount;
383     UINT32                          Interrupts[1];
384     /* ResSourceIndex, ResSource optional fields follow */

386 } AML_RESOURCE_EXTENDED_IRQ;


389 typedef struct aml_resource_generic_register
390 {
```

```
391     AML_RESOURCE_LARGE_HEADER_COMMON
392     UINT8                           AddressSpaceId;
393     UINT8                           BitWidth;
394     UINT8                           BitOffset;
395     UINT8                           AccessSize; /* ACPI 3.0, was previously Rese
396     UINT64                          Address;

398 } AML_RESOURCE_GENERIC_REGISTER;


401 /* Common descriptor for GpioInt and GpioIo (ACPI 5.0) */

403 typedef struct aml_resource_gpio
404 {
405     AML_RESOURCE_LARGE_HEADER_COMMON
406     UINT8                           RevisionId;
407     UINT8                           ConnectionType;
408     UINT16                          Flags;
409     UINT16                          IntFlags;
410     UINT8                           PinConfig;
411     UINT16                          DriveStrength;
412     UINT16                          DebounceTimeout;
413     UINT16                          PinTableOffset;
414     UINT8                           ResSourceIndex;
415     UINT16                          ResSourceOffset;
416     UINT16                          VendorOffset;
417     UINT16                          VendorLength;
418     /*
419      * Optional fields follow immediately:
420      * 1) PIN list (Words)
421      * 2) Resource Source String
422      * 3) Vendor Data bytes
423      */

425 } AML_RESOURCE_GPIO;

427 #define AML_RESOURCE_GPIO_REVISION              1       /* ACPI 5.0 */

429 /* Values for ConnectionType above */

431 #define AML_RESOURCE_GPIO_TYPE_INT              0
432 #define AML_RESOURCE_GPIO_TYPE_IO               1
433 #define AML_RESOURCE_MAX_GPIOTYPE               1


436 /* Common preamble for all serial descriptors (ACPI 5.0) */

438 #define AML_RESOURCE_SERIAL_COMMON \
439     UINT8                           RevisionId; \
440     UINT8                           ResSourceIndex; \
441     UINT8                           Type; \
442     UINT8                           Flags; \
443     UINT16                          TypeSpecificFlags; \
444     UINT8                           TypeRevisionId; \
445     UINT16                          TypeDataLength; \

447 /* Values for the type field above */

449 #define AML_RESOURCE_I2C_SERIALBUSTYPE          1
450 #define AML_RESOURCE_SPI_SERIALBUSTYPE          2
451 #define AML_RESOURCE_UART_SERIALBUSTYPE         3
452 #define AML_RESOURCE_MAX_SERIALBUSTYPE          3
453 #define AML_RESOURCE_VENDOR_SERIALBUSTYPE       192 /* Vendor defined is 0xC0-0x

455 typedef struct aml_resource_common_serialbus
456 {
```



```
457     AML_RESOURCE_LARGE_HEADER_COMMON
458     AML_RESOURCE_SERIAL_COMMON

460 } AML_RESOURCE_COMMON_SERIALBUS;

462 typedef struct aml_resource_i2c_serialbus
463 {
464     AML_RESOURCE_LARGE_HEADER_COMMON
465     AML_RESOURCE_SERIAL_COMMON
466     UINT32                          ConnectionSpeed;
467     UINT16                          SlaveAddress;
468     /*
469      * Optional fields follow immediately:
470      * 1) Vendor Data bytes
471      * 2) Resource Source String
472      */

474 } AML_RESOURCE_I2C_SERIALBUS;

476 #define AML_RESOURCE_I2C_REVISION               1       /* ACPI 5.0 */
477 #define AML_RESOURCE_I2C_TYPE_REVISION          1       /* ACPI 5.0 */
478 #define AML_RESOURCE_I2C_MIN_DATA_LEN           6

480 typedef struct aml_resource_spi_serialbus
481 {
482     AML_RESOURCE_LARGE_HEADER_COMMON
483     AML_RESOURCE_SERIAL_COMMON
484     UINT32                          ConnectionSpeed;
485     UINT8                           DataBitLength;
486     UINT8                           ClockPhase;
487     UINT8                           ClockPolarity;
488     UINT16                          DeviceSelection;
489     /*
490      * Optional fields follow immediately:
491      * 1) Vendor Data bytes
492      * 2) Resource Source String
493      */

495 } AML_RESOURCE_SPI_SERIALBUS;

497 #define AML_RESOURCE_SPI_REVISION               1       /* ACPI 5.0 */
498 #define AML_RESOURCE_SPI_TYPE_REVISION          1       /* ACPI 5.0 */
499 #define AML_RESOURCE_SPI_MIN_DATA_LEN           9


502 typedef struct aml_resource_uart_serialbus
503 {
504     AML_RESOURCE_LARGE_HEADER_COMMON
505     AML_RESOURCE_SERIAL_COMMON
506     UINT32                          DefaultBaudRate;
507     UINT16                          RxFifoSize;
508     UINT16                          TxFifoSize;
509     UINT8                           Parity;
510     UINT8                           LinesEnabled;
511     /*
512      * Optional fields follow immediately:
513      * 1) Vendor Data bytes
514      * 2) Resource Source String
515      */

517 } AML_RESOURCE_UART_SERIALBUS;

519 #define AML_RESOURCE_UART_REVISION              1       /* ACPI 5.0 */
520 #define AML_RESOURCE_UART_TYPE_REVISION         1       /* ACPI 5.0 */
521 #define AML_RESOURCE_UART_MIN_DATA_LEN          10
```

```
524 /* restore default alignment */

526 #pragma pack()

528 /* Union of all resource descriptors, so we can allocate the worst case */

530 typedef union aml_resource
531 {
532     /* Descriptor headers */

534     UINT8                               DescriptorType;
535     AML_RESOURCE_SMALL_HEADER           SmallHeader;
536     AML_RESOURCE_LARGE_HEADER           LargeHeader;

538     /* Small resource descriptors */

540     AML_RESOURCE_IRQ                    Irq;
541     AML_RESOURCE_DMA                    Dma;
542     AML_RESOURCE_START_DEPENDENT        StartDpf;
543     AML_RESOURCE_END_DEPENDENT          EndDpf;
544     AML_RESOURCE_IO                     Io;
545     AML_RESOURCE_FIXED_IO               FixedIo;
546     AML_RESOURCE_FIXED_DMA              FixedDma;
547     AML_RESOURCE_VENDOR_SMALL           VendorSmall;
548     AML_RESOURCE_END_TAG                EndTag;

550     /* Large resource descriptors */

552     AML_RESOURCE_MEMORY24               Memory24;
553     AML_RESOURCE_GENERIC_REGISTER       GenericReg;
554     AML_RESOURCE_VENDOR_LARGE           VendorLarge;
555     AML_RESOURCE_MEMORY32               Memory32;
556     AML_RESOURCE_FIXED_MEMORY32         FixedMemory32;
557     AML_RESOURCE_ADDRESS16              Address16;
558     AML_RESOURCE_ADDRESS32              Address32;
559     AML_RESOURCE_ADDRESS64              Address64;
560     AML_RESOURCE_EXTENDED_ADDRESS64     ExtAddress64;
561     AML_RESOURCE_EXTENDED_IRQ           ExtendedIrq;
562     AML_RESOURCE_GPIO                   Gpio;
563     AML_RESOURCE_I2C_SERIALBUS          I2cSerialBus;
564     AML_RESOURCE_SPI_SERIALBUS          SpiSerialBus;
565     AML_RESOURCE_UART_SERIALBUS         UartSerialBus;
566     AML_RESOURCE_COMMON_SERIALBUS       CommonSerialBus;

568     /* Utility overlays */

570     AML_RESOURCE_ADDRESS                Address;
571     UINT32                              DwordItem;
572     UINT16                              WordItem;
573     UINT8                               ByteItem;

575 } AML_RESOURCE;

577 #endif
```

```
**********************************************************
    3547 Thu Dec 26 13:49:59 2013
new/usr/src/common/acpica/include/platform/accygwin.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: accygwin.h - OS specific defines, etc. for cygwin environment
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACCYGWIN_H__
#define __ACCYGWIN_H__

/*
 * ACPICA configuration
 */
#define ACPI_USE_SYSTEM_CLIBRARY
#define ACPI_USE_DO_WHILE_0
#define ACPI_FLUSH_CPU_CACHE()

/*
 * This is needed since sem_timedwait does not appear to work properly
 * on cygwin (always hangs forever).
 */
#define ACPI_USE_ALTERNATE_TIMEOUT
```



```
#include <stdarg.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>

#if defined(__ia64__) || defined(__x86_64__)
#define ACPI_MACHINE_WIDTH          64
#define COMPILER_DEPENDENT_INT64    long
#define COMPILER_DEPENDENT_UINT64   unsigned long
#else
#define ACPI_MACHINE_WIDTH          32
#define COMPILER_DEPENDENT_INT64    long long
#define COMPILER_DEPENDENT_UINT64   unsigned long long
#define ACPI_USE_NATIVE_DIVIDE
#endif

#ifndef __cdecl
#define __cdecl
#endif

#define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq) if (GLptr) Acq=1; else Acq=0;
#define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pending) Pending = 1

/* On Cygwin, pthread_t is a pointer */

#define ACPI_CAST_PTHREAD_T(pthread) ((ACPI_THREAD_ID) ACPI_TO_INTEGER (pthread)

/* Cygwin uses GCC */

#include "acgcc.h"

/*
 * The vsnprintf function is defined by c99, but cygwin/gcc does not
 * enable this prototype when the -ansi flag is set. Also related to
 * __STRICT_ANSI__. So, we just declare the prototype here.
 */
int
vsnprintf (char *s, size_t n, const char *format, va_list ap);

#endif /* __ACCYGWIN_H__ */
```

```
**********************************************************
    2836 Thu Dec 26 13:50:00 2013
new/usr/src/common/acpica/include/platform/acefi.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acefi.h - OS specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACEFI_H__
#define __ACEFI_H__

#include <efi.h>
#include <efistdarg.h>
#include <efilib.h>


/* _int64 works for both IA32 and IA64 */

#define COMPILER_DEPENDENT_INT64   __int64
#define COMPILER_DEPENDENT_UINT64  unsigned __int64

/*
 * Calling conventions:
 *
```



```
 * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
 * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
 * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
 * ACPI_INTERNAL_VAR_XFACE  - Internal variable-parameter list interfaces
 */
#define ACPI_SYSTEM_XFACE
#define ACPI_EXTERNAL_XFACE
#define ACPI_INTERNAL_XFACE
#define ACPI_INTERNAL_VAR_XFACE

/* warn C4142: redefinition of type */

#pragma warning(disable:4142)


#endif /* __ACEFI_H__ */
```

```
**********************************************************
   12451 Thu Dec 26 13:50:00 2013
new/usr/src/common/acpica/include/platform/acenv.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acenv.h - Host and compiler configuration
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACENV_H__
  45 #define __ACENV_H__

  47 /*
  48  * Environment configuration. The purpose of this file is to interface ACPICA
  49  * to the local environment. This includes compiler-specific, OS-specific,
  50  * and machine-specific configuration.
  51  */

  53 /* Types for ACPI_MUTEX_TYPE */

  55 #define ACPI_BINARY_SEMAPHORE       0
  56 #define ACPI_OSL_MUTEX              1

  58 /* Types for DEBUGGER_THREADING */
```



```
  60 #define DEBUGGER_SINGLE_THREADED    0
  61 #define DEBUGGER_MULTI_THREADED     1


  64 /******************************************************************************
  65  *
  66  * Configuration for ACPI tools and utilities
  67  *
  68  *****************************************************************************/

  70 /* iASL configuration */

  72 #ifdef ACPI_ASL_COMPILER
  73 #define ACPI_APPLICATION
  74 #define ACPI_DISASSEMBLER
  75 #define ACPI_DEBUG_OUTPUT
  76 #define ACPI_CONSTANT_EVAL_ONLY
  77 #define ACPI_LARGE_NAMESPACE_NODE
  78 #define ACPI_DATA_TABLE_DISASSEMBLY
  79 #define ACPI_SINGLE_THREADED
  80 #endif

  82 /* AcpiExec configuration. Multithreaded with full AML debugger */
  82 /* AcpiExec and AcpiBin configuration */

  84 #ifdef ACPI_EXEC_APP
  85 #define ACPI_APPLICATION
  86 #define ACPI_FULL_DEBUG
  87 #define ACPI_MUTEX_DEBUG
  88 #define ACPI_DBG_TRACK_ALLOCATIONS
  89 #endif

  91 /* AcpiNames configuration. Single threaded with debugger output enabled. */

  93 #ifdef ACPI_NAMES_APP
  94 #define ACPI_DEBUGGER
  91 #ifdef ACPI_BIN_APP
  95 #define ACPI_APPLICATION
  96 #define ACPI_SINGLE_THREADED
  97 #endif

  99 /*
 100  * AcpiBin/AcpiDump/AcpiSrc/AcpiXtract/Example configuration. All single
 101  * threaded, with no debug output.
 102  */
 103 #if (defined ACPI_BIN_APP)      || \
 104     (defined ACPI_DUMP_APP)     || \
 105     (defined ACPI_SRC_APP)      || \
 106     (defined ACPI_XTRACT_APP)   || \
 107     (defined ACPI_EXAMPLE_APP)
 108 #define ACPI_APPLICATION
 109 #define ACPI_SINGLE_THREADED
 110 #endif

 112 #ifdef ACPI_HELP_APP
 113 #define ACPI_APPLICATION
 114 #define ACPI_SINGLE_THREADED
 115 #define ACPI_NO_ERROR_MESSAGES
 116 #endif

 118 /* Linkable ACPICA library */

 120 #ifdef ACPI_LIBRARY
 121 #define ACPI_USE_LOCAL_CACHE
 122 #endif
```

```
 124 /* Common for all ACPICA applications */

 126 #ifdef ACPI_APPLICATION
 127 #define ACPI_USE_SYSTEM_CLIBRARY
 128 #define ACPI_USE_LOCAL_CACHE
 129 #endif

 131 /* Common debug support */

 133 #ifdef ACPI_FULL_DEBUG
 134 #define ACPI_DEBUGGER
 135 #define ACPI_DEBUG_OUTPUT
 136 #define ACPI_DISASSEMBLER
 137 #endif


 140 /*! [Begin] no source code translation */

 142 /******************************************************************************
 143  *
 144  * Host configuration files. The compiler configuration files are included
 145  * by the host files.
 146  *
 147  *****************************************************************************/

 149 #if defined(_LINUX) || defined(__linux__)
 150 #include "aclinux.h"

 152 #elif defined(_APPLE) || defined(__APPLE__)
 153 #include "acmacosx.h"

 155 #elif defined(__FreeBSD__) || defined(__FreeBSD_kernel__)
 156 #include "acfreebsd.h"

 158 #elif defined(__NetBSD__)
 159 #include "acnetbsd.h"

 161 #elif defined(__sun)
 162 #include "acsolaris.h"

 164 #elif defined(MODESTO)
 165 #include "acmodesto.h"

 167 #elif defined(NETWARE)
 168 #include "acnetware.h"

 170 #elif defined(_CYGWIN)
 171 #include "accygwin.h"

 173 #elif defined(WIN32)
 174 #include "acwin.h"

 176 #elif defined(WIN64)
 177 #include "acwin64.h"

 179 #elif defined(_WRS_LIB_BUILD)
 180 #include "acvxworks.h"

 182 #elif defined(__OS2__)
 183 #include "acos2.h"

 185 #elif defined(_AED_EFI)
 186 #include "acefi.h"

 188 #elif defined(__HAIKU__)
```



```
 189 #include "achaiku.h"

 191 #else

 193 /* Unknown environment */

 195 #error Unknown target environment
 196 #endif

 198 /*! [End] no source code translation !*/


 201 /******************************************************************************
 202  *
 203  * Setup defaults for the required symbols that were not defined in one of
 204  * the host/compiler files above.
 205  *
 206  *****************************************************************************/

 208 /* 64-bit data types */

 210 #ifndef COMPILER_DEPENDENT_INT64
 211 #define COMPILER_DEPENDENT_INT64   long long
 212 #endif

 214 #ifndef COMPILER_DEPENDENT_UINT64
 215 #define COMPILER_DEPENDENT_UINT64  unsigned long long
 216 #endif

 218 /* Type of mutex supported by host. Default is binary semaphores. */

 220 #ifndef ACPI_MUTEX_TYPE
 221 #define ACPI_MUTEX_TYPE             ACPI_BINARY_SEMAPHORE
 222 #endif

 224 /* Global Lock acquire/release */

 226 #ifndef ACPI_ACQUIRE_GLOBAL_LOCK
 227 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acquired) Acquired = 1
 228 #endif

 230 #ifndef ACPI_RELEASE_GLOBAL_LOCK
 231 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pending) Pending = 0
 232 #endif

 234 /* Flush CPU cache - used when going to sleep. Wbinvd or similar. */

 236 #ifndef ACPI_FLUSH_CPU_CACHE
 237 #define ACPI_FLUSH_CPU_CACHE()
 238 #endif

 240 /* "inline" keywords - configurable since inline is not standardized */

 242 #ifndef ACPI_INLINE
 243 #define ACPI_INLINE
 244 #endif

 246 /*
 247  * Configurable calling conventions:
 248  *
 249  * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
 250  * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
 251  * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
 252  * ACPI_INTERNAL_VAR_XFACE  - Internal variable-parameter list interfaces
 253  */
 254 #ifndef ACPI_SYSTEM_XFACE
```

```
255 #define ACPI_SYSTEM_XFACE
256 #endif

258 #ifndef ACPI_EXTERNAL_XFACE
259 #define ACPI_EXTERNAL_XFACE
260 #endif

262 #ifndef ACPI_INTERNAL_XFACE
263 #define ACPI_INTERNAL_XFACE
264 #endif

266 #ifndef ACPI_INTERNAL_VAR_XFACE
267 #define ACPI_INTERNAL_VAR_XFACE
268 #endif

270 /*
271  * Debugger threading model
272  * Use single threaded if the entire subsystem is contained in an application
273  * Use multiple threaded when the subsystem is running in the kernel.
274  *
275  * By default the model is single threaded if ACPI_APPLICATION is set,
276  * multi-threaded if ACPI_APPLICATION is not set.
277  */
278 #ifndef DEBUGGER_THREADING
279 #ifdef ACPI_APPLICATION
280 #define DEBUGGER_THREADING          DEBUGGER_SINGLE_THREADED

282 #else
283 #define DEBUGGER_THREADING          DEBUGGER_MULTI_THREADED
284 #endif
285 #endif /* !DEBUGGER_THREADING */


288 /******************************************************************************
289  *
290  * C library configuration
291  *
292  *****************************************************************************/

294 /*
295  * ACPI_USE_SYSTEM_CLIBRARY - Define this if linking to an actual C library.
296  *      Otherwise, local versions of string/memory functions will be used.
297  * ACPI_USE_STANDARD_HEADERS - Define this if linking to a C library and
298  *      the standard header files may be used.
299  *
300  * The ACPICA subsystem only uses low level C library functions that do not call
301  * operating system services and may therefore be inlined in the code.
302  *
303  * It may be necessary to tailor these include files to the target
304  * generation environment.
305  */
306 #ifdef ACPI_USE_SYSTEM_CLIBRARY

308 /* Use the standard C library headers. We want to keep these to a minimum. */
280 /* Use the standard C library headers. We want to keep these to a minimum */

310 #ifdef ACPI_USE_STANDARD_HEADERS

312 /* Use the standard headers from the standard locations */

314 #include <stdarg.h>
315 #include <stdlib.h>
316 #include <string.h>
317 #include <ctype.h>

319 #endif /* ACPI_USE_STANDARD_HEADERS */
```



```
321 /* We will be linking to the standard Clib functions */

323 #define ACPI_STRSTR(s1,s2)      strstr((s1), (s2))
324 #define ACPI_STRCHR(s1,c)       strchr((s1), (c))
325 #define ACPI_STRLEN(s)          (ACPI_SIZE) strlen((s))
326 #define ACPI_STRCPY(d,s)        (void) strcpy((d), (s))
327 #define ACPI_STRNCPY(d,s,n)     (void) strncpy((d), (s), (ACPI_SIZE)(n))
328 #define ACPI_STRNCMP(d,s,n)     strncmp((d), (s), (ACPI_SIZE)(n))
329 #define ACPI_STRCMP(d,s)        strcmp((d), (s))
330 #define ACPI_STRCAT(d,s)        (void) strcat((d), (s))
331 #define ACPI_STRNCAT(d,s,n)     strncat((d), (s), (ACPI_SIZE)(n))
332 #define ACPI_STRTOUL(d,s,n)     strtoul((d), (s), (ACPI_SIZE)(n))
333 #define ACPI_MEMCMP(s1,s2,n)    memcmp((const char *)(s1), (const char *)(s2), (
334 #define ACPI_MEMCPY(d,s,n)      (void) memcpy((d), (s), (ACPI_SIZE)(n))
335 #define ACPI_MEMSET(d,s,n)      (void) memset((d), (s), (ACPI_SIZE)(n))
336 #define ACPI_TOUPPER(i)         toupper((int) (i))
337 #define ACPI_TOLOWER(i)         tolower((int) (i))
338 #define ACPI_IS_XDIGIT(i)       isxdigit((int) (i))
339 #define ACPI_IS_DIGIT(i)        isdigit((int) (i))
340 #define ACPI_IS_SPACE(i)        isspace((int) (i))
341 #define ACPI_IS_UPPER(i)        isupper((int) (i))
342 #define ACPI_IS_PRINT(i)        isprint((int) (i))
343 #define ACPI_IS_ALPHA(i)        isalpha((int) (i))

345 #else

347 /******************************************************************************
348  *
349  * Not using native C library, use local implementations
350  *
351  *****************************************************************************/

353 /*
354  * Use local definitions of C library macros and functions. These function
355  * implementations may not be as efficient as an inline or assembly code
356  * implementation provided by a native C library, but they are functionally
357  * equivalent.
358  */
359 #ifndef va_arg

361 #ifndef _VALIST
362 #define _VALIST
363 typedef char *va_list;
364 #endif /* _VALIST */

366 /* Storage alignment properties */

368 #define  _AUPBND                (sizeof (ACPI_NATIVE_INT) - 1)
369 #define  _ADNBND                (sizeof (ACPI_NATIVE_INT) - 1)

371 /* Variable argument list macro definitions */

373 #define _Bnd(X, bnd)            (((sizeof (X)) + (bnd)) & (~(bnd)))
374 #define va_arg(ap, T)           (*(T *)(((ap) += (_Bnd (T, _AUPBND))) - (_Bnd (T
375 #define va_end(ap)              (ap = (va_list) NULL)
347 #define va_end(ap)              (void) 0
376 #define va_start(ap, A)         (void) ((ap) = (((char *) &(A)) + (_Bnd (A,_AUPB

378 #endif /* va_arg */

380 /* Use the local (ACPICA) definitions of the clib functions */

382 #define ACPI_STRSTR(s1,s2)      AcpiUtStrstr ((s1), (s2))
383 #define ACPI_STRCHR(s1,c)       AcpiUtStrchr ((s1), (c))
384 #define ACPI_STRLEN(s)          (ACPI_SIZE) AcpiUtStrlen ((s))
```

```
385 #define ACPI_STRCPY(d,s)        (void) AcpiUtStrcpy ((d), (s))
386 #define ACPI_STRNCPY(d,s,n)     (void) AcpiUtStrncpy ((d), (s), (ACPI_SIZE)(n))
387 #define ACPI_STRNCMP(d,s,n)     AcpiUtStrncmp ((d), (s), (ACPI_SIZE)(n))
388 #define ACPI_STRCMP(d,s)        AcpiUtStrcmp ((d), (s))
389 #define ACPI_STRCAT(d,s)        (void) AcpiUtStrcat ((d), (s))
390 #define ACPI_STRNCAT(d,s,n)     AcpiUtStrncat ((d), (s), (ACPI_SIZE)(n))
391 #define ACPI_STRTOUL(d,s,n)     AcpiUtStrtoul ((d), (s), (ACPI_SIZE)(n))
392 #define ACPI_MEMCMP(s1,s2,n)    AcpiUtMemcmp((const char *)(s1), (const char *)(
393 #define ACPI_MEMCPY(d,s,n)      (void) AcpiUtMemcpy ((d), (s), (ACPI_SIZE)(n))
394 #define ACPI_MEMSET(d,v,n)      (void) AcpiUtMemset ((d), (v), (ACPI_SIZE)(n))
395 #define ACPI_TOUPPER(c)         AcpiUtToUpper ((int) (c))
396 #define ACPI_TOLOWER(c)         AcpiUtToLower ((int) (c))

398 #endif /* ACPI_USE_SYSTEM_CLIBRARY */

400 #ifndef ACPI_FILE
401 #ifdef ACPI_APPLICATION
402 #include <stdio.h>
403 #define ACPI_FILE               FILE *
404 #else
405 #define ACPI_FILE               void *
406 #endif /* ACPI_APPLICATION */
407 #endif /* ACPI_FILE */

409 #endif /* __ACENV_H__ */
```

```
**********************************************************
    3487 Thu Dec 26 13:50:00 2013
new/usr/src/common/acpica/include/platform/acfreebsd.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acfreebsd.h - OS specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACFREEBSD_H__
#define __ACFREEBSD_H__


/* FreeBSD uses GCC */

#include "acgcc.h"
#include <sys/types.h>
#include <machine/acpica_machdep.h>

#ifdef __LP64__
#define ACPI_MACHINE_WIDTH      64
#else
#define ACPI_MACHINE_WIDTH      32
#endif
```



```
#define COMPILER_DEPENDENT_INT64        int64_t
#define COMPILER_DEPENDENT_UINT64       uint64_t

#define ACPI_UINTPTR_T      uintptr_t

#define ACPI_USE_DO_WHILE_0
#define ACPI_USE_LOCAL_CACHE
#define ACPI_USE_NATIVE_DIVIDE
#define ACPI_USE_SYSTEM_CLIBRARY

#ifdef _KERNEL

#include <sys/ctype.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/libkern.h>
#include <machine/acpica_machdep.h>
#include <machine/stdarg.h>

#include "opt_acpi.h"

#define ACPI_MUTEX_TYPE     ACPI_OSL_MUTEX

#ifdef ACPI_DEBUG
#define ACPI_DEBUG_OUTPUT   /* for backward compatibility */
#define ACPI_DISASSEMBLER
#endif

#ifdef ACPI_DEBUG_OUTPUT
#include "opt_ddb.h"
#ifdef DDB
#define ACPI_DEBUGGER
#endif /* DDB */
#endif /* ACPI_DEBUG_OUTPUT */

#ifdef DEBUGGER_THREADING
#undef DEBUGGER_THREADING
#endif /* DEBUGGER_THREADING */

#define DEBUGGER_THREADING  0   /* integrated with DDB */

#else /* _KERNEL */

#if __STDC_HOSTED__
#include <ctype.h>
#endif

#define ACPI_CAST_PTHREAD_T(pthread)    ((ACPI_THREAD_ID) ACPI_TO_INTEGER (pthre

#define ACPI_USE_STANDARD_HEADERS

#define ACPI_FLUSH_CPU_CACHE()
#define __cdecl

#endif /* _KERNEL */

#endif /* __ACFREEBSD_H__ */
```

****
2843 Thu Dec 26 13:50:01 2013
new/usr/src/common/acpica/include/platform/acgcc.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acgcc.h - GCC specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACGCC_H__
#define __ACGCC_H__

#define ACPI_INLINE             __inline__

/* Function name is used for debug output. Non-ANSI, compiler-dependent */

#define ACPI_GET_FUNCTION_NAME          __func__
#define ACPI_GET_FUNCTION_NAME          __FUNCTION__

/*
 * This macro is used to tag functions as "printf-like" because
 * some compilers (like GCC) can catch printf format string problems.
 */
#define ACPI_PRINTF_LIKE(c) __attribute__ ((__format__ (__printf__, c, c+1)))
```



```
/*
 * Some compilers complain about unused variables. Sometimes we don't want to
 * use all the variables (for example, _AcpiModuleName). This allows us
 * to tell the compiler warning in a per-variable manner that a variable
 * to to tell the compiler warning in a per-variable manner that a variable
 * is unused.
 */
#define ACPI_UNUSED_VAR __attribute__ ((unused))

#endif /* __ACGCC_H__ */
```

```
**********************************************************
    3764 Thu Dec 26 13:50:01 2013
new/usr/src/common/acpica/include/platform/achaiku.h
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Name: achaiku.h - OS specific defines, etc. for Haiku (www.haiku-os.org)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACHAIKU_H__
#define __ACHAIKU_H__

#include "acgcc.h"
#include <KernelExport.h>

struct mutex;


/* Host-dependent types and defines for user- and kernel-space ACPICA */

#define ACPI_USE_SYSTEM_CLIBRARY
#define ACPI_USE_STANDARD_HEADERS

#define ACPI_MUTEX_TYPE             ACPI_OSL_MUTEX
#define ACPI_MUTEX                  struct mutex *

#define ACPI_USE_NATIVE_DIVIDE
```



```
// #define ACPI_THREAD_ID              thread_id
#define ACPI_SEMAPHORE              sem_id
#define ACPI_SPINLOCK               spinlock *
#define ACPI_CPU_FLAGS              cpu_status

#define COMPILER_DEPENDENT_INT64    int64
#define COMPILER_DEPENDENT_UINT64   uint64


#ifdef B_HAIKU_64_BIT
#define ACPI_MACHINE_WIDTH          64
#else
#define ACPI_MACHINE_WIDTH          32
#endif


#ifdef _KERNEL_MODE
/* Host-dependent types and defines for in-kernel ACPICA */

/* ACPICA cache implementation is adequate. */
#define ACPI_USE_LOCAL_CACHE

#define ACPI_FLUSH_CPU_CACHE() __asm __volatile("wbinvd");

/* Based on FreeBSD's due to lack of documentation */
extern int AcpiOsAcquireGlobalLock(uint32 *lock);
extern int AcpiOsReleaseGlobalLock(uint32 *lock);

#define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)    do {                \
        (Acq) = AcpiOsAcquireGlobalLock(&((GLptr)->GlobalLock));    \
} while (0)

#define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Acq)    do {                \
        (Acq) = AcpiOsReleaseGlobalLock(&((GLptr)->GlobalLock));    \
} while (0)

#else /* _KERNEL_MODE */
/* Host-dependent types and defines for user-space ACPICA */

#error "We only support kernel mode ACPI atm."

#endif /* _KERNEL_MODE */
#endif /* __ACHAIKU_H__ */
```

****
3196 Thu Dec 26 13:50:01 2013
new/usr/src/common/acpica/include/platform/acintel.h
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acintel.h - VC specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACINTEL_H__
#define __ACINTEL_H__

/* Configuration specific to Intel 64-bit C compiler */

#define COMPILER_DEPENDENT_INT64    __int64
#define COMPILER_DEPENDENT_UINT64   unsigned __int64
#define ACPI_INLINE                 __inline

/*
 * Calling conventions:
 *
 * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
 * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
 * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
 * ACPI_INTERNAL_VAR_XFACE  - Internal variable-parameter list interfaces
```



```
 */
#define ACPI_SYSTEM_XFACE
#define ACPI_EXTERNAL_XFACE
#define ACPI_INTERNAL_XFACE
#define ACPI_INTERNAL_VAR_XFACE

/* remark 981 - operands evaluated in no particular order */
#pragma warning(disable:981)

/* warn C4100: unreferenced formal parameter */
#pragma warning(disable:4100)

/* warn C4127: conditional expression is constant */
#pragma warning(disable:4127)

/* warn C4706: assignment within conditional expression */
#pragma warning(disable:4706)

/* warn C4214: bit field types other than int */
#pragma warning(disable:4214)


#endif /* __ACINTEL_H__ */
```

```
**********************************************************
    7957 Thu Dec 26 13:50:02 2013
new/usr/src/common/acpica/include/platform/aclinux.h
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: aclinux.h - OS specific defines, etc. for Linux
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACLINUX_H__
#define __ACLINUX_H__

/* Common (in-kernel/user-space) ACPICA configuration */

#define ACPI_USE_SYSTEM_CLIBRARY
#define ACPI_USE_DO_WHILE_0
#define ACPI_MUTEX_TYPE             ACPI_BINARY_SEMAPHORE


#ifdef __KERNEL__

#include <linux/string.h>
#include <linux/kernel.h>
```



```
#include <linux/ctype.h>
#include <linux/sched.h>
#include <linux/atomic.h>
#include <linux/math64.h>
#include <linux/slab.h>
#include <linux/spinlock_types.h>
#ifdef EXPORT_ACPI_INTERFACES
#include <linux/export.h>
#endif
#include <asm/acpi.h>

/* Host-dependent types and defines for in-kernel ACPICA */

#define ACPI_MACHINE_WIDTH          BITS_PER_LONG
#define ACPI_EXPORT_SYMBOL(symbol)  EXPORT_SYMBOL(symbol);
#define strtoul                     simple_strtoul

#define ACPI_CACHE_T                struct kmem_cache
#define ACPI_SPINLOCK               spinlock_t *
#define ACPI_CPU_FLAGS              unsigned long

#else /* !__KERNEL__ */

#include <stdarg.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>

/* Host-dependent types and defines for user-space ACPICA */

#define ACPI_FLUSH_CPU_CACHE()
#define ACPI_CAST_PTHREAD_T(Pthread) ((ACPI_THREAD_ID) (Pthread))

#if defined(__ia64__) || defined(__x86_64__) || defined(__aarch64__)
#define ACPI_MACHINE_WIDTH          64
#define COMPILER_DEPENDENT_INT64    long
#define COMPILER_DEPENDENT_UINT64   unsigned long
#else
#define ACPI_MACHINE_WIDTH          32
#define COMPILER_DEPENDENT_INT64    long long
#define COMPILER_DEPENDENT_UINT64   unsigned long long
#define ACPI_USE_NATIVE_DIVIDE
#endif

#ifndef __cdecl
#define __cdecl
#endif

#endif /* __KERNEL__ */

/* Linux uses GCC */

#include "acgcc.h"


#ifdef __KERNEL__

/*
 * FIXME: Inclusion of actypes.h
 * Linux kernel need this before defining inline OSL interfaces as
 * actypes.h need to be included to find ACPICA type definitions.
 * Since from ACPICA's perspective, the actypes.h should be included after
 * acenv.h (aclinux.h), this leads to a inclusion mis-ordering issue.
 */
#include <acpi/actypes.h>
```

```
125 /*
126  * Overrides for in-kernel ACPICA
127  */
128 ACPI_STATUS __init AcpiOsInitialize (
129     void);
130 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsInitialize

132 ACPI_STATUS AcpiOsTerminate (
133     void);
134 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsTerminate

136 /*
137  * Memory allocation/deallocation
138  */

140 /*
141  * The irqs_disabled() check is for resume from RAM.
142  * Interrupts are off during resume, just like they are for boot.
143  * However, boot has  (system_state != SYSTEM_RUNNING)
144  * to quiet __might_sleep() in kmalloc() and resume does not.
145  */
146 static inline void *
147 AcpiOsAllocate (
148     ACPI_SIZE               Size)
149 {
150     return kmalloc (Size, irqs_disabled () ? GFP_ATOMIC : GFP_KERNEL);
151 }
152 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAllocate

154 /* Use native linux version of AcpiOsAllocateZeroed */

156 static inline void *
157 AcpiOsAllocateZeroed (
158     ACPI_SIZE               Size)
159 {
160     return kzalloc (Size, irqs_disabled () ? GFP_ATOMIC : GFP_KERNEL);
161 }
162 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAllocateZeroed
163 #define USE_NATIVE_ALLOCATE_ZEROED

165 static inline void
166 AcpiOsFree (
167     void                    *Memory)
168 {
169     kfree (Memory);
170 }
171 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsFree

173 static inline void *
174 AcpiOsAcquireObject (
175     ACPI_CACHE_T            *Cache)
176 {
177     return kmem_cache_zalloc (Cache,
178         irqs_disabled () ? GFP_ATOMIC : GFP_KERNEL);
179 }
180 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsAcquireObject

182 static inline ACPI_THREAD_ID
183 AcpiOsGetThreadId (
184     void)
185 {
186     return (ACPI_THREAD_ID) (unsigned long) current;
187 }
188 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetThreadId
```



```
190 #ifndef CONFIG_PREEMPT

192 /*
193  * Used within ACPICA to show where it is safe to preempt execution
194  * when CONFIG_PREEMPT=n
195  */
196 #define ACPI_PREEMPTION_POINT() \
197     do { \
198         if (!irqs_disabled()) \
199             cond_resched(); \
200     } while (0)

202 #endif

204 /*
205  * When lockdep is enabled, the spin_lock_init() macro stringifies it's
206  * argument and uses that as a name for the lock in debugging.
207  * By executing spin_lock_init() in a macro the key changes from "lock" for
208  * all locks to the name of the argument of acpi_os_create_lock(), which
209  * prevents lockdep from reporting false positives for ACPICA locks.
210  */
211 #define AcpiOsCreateLock(__Handle) \
212     ({ \
213         spinlock_t *Lock = ACPI_ALLOCATE(sizeof(*Lock)); \
214         if (Lock) { \
215             *(__Handle) = Lock; \
216             spin_lock_init(*(__Handle)); \
217         } \
218         Lock ? AE_OK : AE_NO_MEMORY; \
219     })
220 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCreateLock

222 void __iomem *
223 AcpiOsMapMemory (
224     ACPI_PHYSICAL_ADDRESS   Where,
225     ACPI_SIZE               Length);
226 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsMapMemory

228 void
229 AcpiOsUnmapMemory (
230     void __iomem            *LogicalAddress,
231     ACPI_SIZE               Size);
232 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsUnmapMemory

234 /*
235  * OSL interfaces used by debugger/disassembler
236  */
237 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsReadable
238 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsWritable

240 /*
241  * OSL interfaces used by utilities
242  */
243 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsRedirectOutput
244 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetLine
245 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByName
246 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByIndex
247 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetTableByAddress
248 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsOpenDirectory
249 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetNextFilename
250 #define ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsCloseDirectory

252 /*
253  * OSL interfaces added by Linux
254  */
255 void
```

```
256 EarlyAcpiOsUnmapMemory (
257     void __iomem            *Virt,
258     ACPI_SIZE               Size);

260 ACPI_STATUS
261 AcpiOsHotplugExecute (
262     ACPI_OSD_EXEC_CALLBACK  Function,
263     void                    *Context);

265 #endif /* __KERNEL__ */

267 #endif /* __ACLINUX_H__ */
```

****
2399 Thu Dec 26 13:50:02 2013
new/usr/src/common/acpica/include/platform/acmacosx.h
acpica-unix2-20130823
****

```
/******************************************************************************
 *
 * Name: acmacosx.h - OS specific defines, etc. for Mac OS X
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACMACOSX_H__
#define __ACMACOSX_H__

#include "aclinux.h"

#ifdef __APPLE__
#define sem_destroy         sem_close
#define ACPI_USE_ALTERNATE_TIMEOUT
#endif /* __APPLE__ */

#ifdef __clang__
#pragma clang diagnostic ignored "-Wformat-nonliteral"
#endif

#endif /* __ACMACOSX_H__ */
```

```
**********************************************************
    5413 Thu Dec 26 13:50:02 2013
new/usr/src/common/acpica/include/platform/acmsvc.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Name: acmsvc.h - VC specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACMSVC_H__
#define __ACMSVC_H__

/*
 * Map low I/O functions for MS. This allows us to disable MS language
 * extensions for maximum portability.
 */
#define open            _open
#define read            _read
#define write           _write
#define close           _close
#define stat            _stat
#define fstat           _fstat
#define mkdir           _mkdir
#define strlwr          _strlwr
```



```
#define O_RDONLY        _O_RDONLY
#define O_BINARY        _O_BINARY
#define O_CREAT         _O_CREAT
#define O_WRONLY        _O_WRONLY
#define O_TRUNC         _O_TRUNC
#define S_IREAD         _S_IREAD
#define S_IWRITE        _S_IWRITE
#define S_IFDIR         _S_IFDIR

/* Eliminate warnings for "old" (non-secure) versions of clib functions */

#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#endif

/* Eliminate warnings for POSIX clib function names (open, write, etc.) */

#ifndef _CRT_NONSTDC_NO_DEPRECATE
#define _CRT_NONSTDC_NO_DEPRECATE
#endif

#define COMPILER_DEPENDENT_INT64    __int64
#define COMPILER_DEPENDENT_UINT64   unsigned __int64
#define ACPI_INLINE                 __inline

/*
 * Calling conventions:
 *
 * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
 * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
 * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
 * ACPI_INTERNAL_VAR_XFACE  - Internal variable-parameter list interfaces
 */
#define ACPI_SYSTEM_XFACE           __cdecl
#define ACPI_EXTERNAL_XFACE
#define ACPI_INTERNAL_XFACE
#define ACPI_INTERNAL_VAR_XFACE     __cdecl

#ifndef _LINT
/*
 * Math helper functions
 */
#define ACPI_DIV_64_BY_32(n_hi, n_lo, d32, q32, r32) \
{                           \
    __asm mov    edx, n_hi  \
    __asm mov    eax, n_lo  \
    __asm div    d32        \
    __asm mov    q32, eax   \
    __asm mov    r32, edx   \
}
_____unchanged_portion_omitted_
#endif

/* warn C4100: unreferenced formal parameter */
#pragma warning(disable:4100)

/* warn C4127: conditional expression is constant */
#pragma warning(disable:4127)

/* warn C4706: assignment within conditional expression */
#pragma warning(disable:4706)

/* warn C4131: uses old-style declarator (iASL compiler only) */
#pragma warning(disable:4131)

#if _MSC_VER > 1200 /* Versions above VC++ 6 */
```

```
#pragma warning( disable : 4295 ) /* needed for acpredef.h array */
#endif

/* Debug support. Must be last in this file, do not move. */

#ifdef _DEBUG
#include <crtdbg.h>

/*
 * Debugging memory corruption issues with windows:
 * Add #include <crtdbg.h> to accommon.h if necessary.
 * Add _ASSERTE(_CrtCheckMemory()); where needed to test memory integrity.
 * This can quickly localize the memory corruption.
 */
#define ACPI_DEBUG_INITIALIZE() \
    _CrtSetDbgFlag (_CRTDBG_CHECK_ALWAYS_DF | \
        _CRTDBG_ALLOC_MEM_DF | _CRTDBG_CHECK_CRT_DF | \
        _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG))
#endif

#endif /* __ACMSVC_H__ */
```

****
3518 Thu Dec 26 13:50:03 2013
new/usr/src/common/acpica/include/platform/acnetbsd.h
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acnetbsd.h - OS specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACNETBSD_H__
#define __ACNETBSD_H__

/* NetBSD uses GCC */

#include "acgcc.h"

#ifdef _LP64
#define ACPI_MACHINE_WIDTH      64
#else
#define ACPI_MACHINE_WIDTH      32
#endif

#define COMPILER_DEPENDENT_INT64  int64_t
#define COMPILER_DEPENDENT_UINT64 uint64_t
```



```
#ifdef _KERNEL
#include "opt_acpi.h"           /* collect build-time options here */

#include <sys/param.h>
#include <sys/systm.h>
#include <machine/stdarg.h>
#include <machine/acpi_func.h>

#define asm         __asm

#define ACPI_USE_NATIVE_DIVIDE

#define ACPI_SYSTEM_XFACE
#define ACPI_EXTERNAL_XFACE
#define ACPI_INTERNAL_XFACE
#define ACPI_INTERNAL_VAR_XFACE

#ifdef ACPI_DEBUG
#define ACPI_DEBUG_OUTPUT
#define ACPI_DBG_TRACK_ALLOCATIONS
#ifdef DEBUGGER_THREADING
#undef DEBUGGER_THREADING
#endif /* DEBUGGER_THREADING */
#define DEBUGGER_THREADING 0    /* integrated with DDB */
#include "opt_ddb.h"
#ifdef DDB
#define ACPI_DISASSEMBLER
#define ACPI_DEBUGGER
#endif /* DDB */
#endif /* ACPI_DEBUG */

static __inline int
isprint(int ch)
{
        return(isspace(ch) || isascii(ch));
}
```

*_____unchanged_portion_omitted_*

****
3565 Thu Dec 26 13:50:03 2013
new/usr/src/common/acpica/include/platform/acos2.h
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acos2.h - OS/2 specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACOS2_H__
#define __ACOS2_H__
#define INCL_LONGLONG
#include <os2.h>


#define ACPI_MACHINE_WIDTH          32

#define COMPILER_DEPENDENT_INT64    long long
#define COMPILER_DEPENDENT_UINT64   unsigned long long
#define ACPI_USE_NATIVE_DIVIDE

#define ACPI_SYSTEM_XFACE           APIENTRY
#define ACPI_EXTERNAL_XFACE         APIENTRY
#define ACPI_INTERNAL_XFACE         APIENTRY
#define ACPI_INTERNAL_VAR_XFACE     APIENTRY
```



```
/*
 * Some compilers complain about unused variables. Sometimes we don't want to
 * use all the variables (most specifically for _THIS_MODULE). This allow us
 * to to tell the compiler warning in a per-variable manner that a variable
 * is unused.
 */
#define ACPI_UNUSED_VAR

#define ACPI_USE_STANDARD_HEADERS
#include <io.h>

#define ACPI_FLUSH_CPU_CACHE() Wbinvd()
void Wbinvd(void);

#define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)       Acq = OSPMAcquireGlobalLock(G
#define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd)       Pnd = OSPMReleaseGlobalLock(G
unsigned short OSPMAcquireGlobalLock (void *);
unsigned short OSPMReleaseGlobalLock (void *);

#define ACPI_SHIFT_RIGHT_64(n_hi, n_lo) \
{ \
    unsigned long long val = 0LL; \
    val = n_lo | ( ((unsigned long long)h_hi) << 32 ); \
    __llrotr (val,1); \
    n_hi = (unsigned long)((val >> 32 ) & 0xffffffff ); \
    n_lo = (unsigned long)(val & 0xffffffff); \
}
```

unchanged_portion_omitted_

```
**********************************************************
    2832 Thu Dec 26 13:50:03 2013
new/usr/src/common/acpica/include/platform/acsolaris.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

#ifndef _ACSOLARIS_H_
#define _ACSOLARIS_H_

#ifdef __cplusplus
extern "C" {
#endif

#ifdef _KERNEL
#include <sys/types.h>
#include <sys/sunddi.h>
#include <sys/varargs.h>
#include <sys/cpu.h>
#include <sys/thread.h>
#else
#include <ctype.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#endif

#ifdef __GNUC__
#include "acgcc.h"
#else
/* Function name used for debug output. */
#define ACPI_GET_FUNCTION_NAME  __func__
#define ACPI_UNUSED_VAR
#endif

uint32_t __acpi_acquire_global_lock(void *);
uint32_t __acpi_release_global_lock(void *);
void     __acpi_wbinvd(void);
```



```
#ifdef  _ILP32
#define ACPI_MACHINE_WIDTH      32
#elif   defined(_LP64)
#define ACPI_MACHINE_WIDTH      64
#endif

#define COMPILER_DEPENDENT_INT64        int64_t
#define COMPILER_DEPENDENT_UINT64       uint64_t

#define ACPI_CAST_PTHREAD_T(pthread)    ((ACPI_THREAD_ID) (pthread))

#define ACPI_PRINTF_LIKE_FUNC

#define ACPI_UNUSED_VAR
#define ACPI_USE_NATIVE_DIVIDE
#define ACPI_FLUSH_CPU_CACHE()  (__acpi_wbinvd())

#ifdef DEBUG
#define ACPI_DEBUG_OUTPUT
#define ACPI_DISASSEMBLER
#endif

#define ACPI_PACKED_POINTERS_NOT_SUPPORTED

/*
 * Calling conventions:
 *
 * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
 * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
 * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
 * ACPI_INTERNAL_VAR_XFACE  - Internal variable-parameter list interfaces
 */
#define ACPI_SYSTEM_XFACE
#define ACPI_EXTERNAL_XFACE
#define ACPI_INTERNAL_XFACE
#define ACPI_INTERNAL_VAR_XFACE

#define ACPI_ASM_MACROS
#define BREAKPOINT3
#define ACPI_DISABLE_IRQS()     cli()
#define ACPI_ENABLE_IRQS()      sti()
#define ACPI_ACQUIRE_GLOBAL_LOCK(Facs, Acq)     \
        ((Acq) = __acpi_acquire_global_lock(Facs))

#define ACPI_RELEASE_GLOBAL_LOCK(Facs, Acq)     \
        ((Acq) = __acpi_release_global_lock(Facs))

#ifdef __cplusplus
}
_____unchanged_portion_omitted_
```

```
**********************************************************
    5683 Thu Dec 26 13:50:04 2013
new/usr/src/common/acpica/include/platform/acwin.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Name: acwin.h - OS specific defines, etc.
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   8  * Copyright (C) 2000 - 2011, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #ifndef __ACWIN_H__
  45 #define __ACWIN_H__

  47 /*! [Begin] no source code translation (Keep the include) */

  49 /* Windows uses VC */
  50 #ifdef _MSC_VER
  51 #include "acmsvc.h"
  52 #endif
  53 /*! [End] no source code translation !*/

  55 #define ACPI_MACHINE_WIDTH      32

  57 #define ACPI_USE_STANDARD_HEADERS

  59 #ifdef ACPI_DEFINE_ALTERNATE_TYPES
```



```
  60 /*
  61  * Types used only in (Linux) translated source, defined here to enable
  62  * cross-platform compilation (i.e., generate the Linux code on Windows,
  63  * for test purposes only)
  64  */
  65 typedef int                             s32;
  66 typedef unsigned char                   u8;
  67 typedef unsigned short                  u16;
  68 typedef unsigned int                    u32;
  69 typedef COMPILER_DEPENDENT_UINT64       u64;
  70 #endif


  73 /*
  74  * Handle platform- and compiler-specific assembly language differences.
  75  *
  76  * Notes:
  77  * 1) Interrupt 3 is used to break into a debugger
  78  * 2) Interrupts are turned off during ACPI register setup
  79  */

  81 /*! [Begin] no source code translation  */

  83 #ifdef ACPI_APPLICATION
  84 #define ACPI_FLUSH_CPU_CACHE()
  85 #else
  86 #define ACPI_FLUSH_CPU_CACHE()  __asm {WBINVD}
  87 #endif

  89 #ifdef _DEBUG
  90 #define ACPI_SIMPLE_RETURN_MACROS
  91 #endif

  93 /*! [End] no source code translation !*/

  95 /*
  96  * Global Lock acquire/release code
  97  *
  98  * Note: Handles case where the FACS pointer is null
  99  */
 100 #define ACPI_ACQUIRE_GLOBAL_LOCK(FacsPtr, Acq)  __asm \
 101 {                                                   \
 102         __asm mov           eax, 0xFF               \
 103         __asm mov           ecx, FacsPtr            \
 104         __asm or            ecx, ecx                \
 105         __asm jz            exit_acq                \
 106         __asm lea           ecx, [ecx].GlobalLock   \
 107                                                     \
 108         __asm acq10:                                \
 109         __asm mov           eax, [ecx]              \
 110         __asm mov           edx, eax                \
 111         __asm and           edx, 0xFFFFFFFE         \
 112         __asm bts           edx, 1                  \
 113         __asm adc           edx, 0                  \
 114         __asm lock cmpxchg  dword ptr [ecx], edx    \
 115         __asm jnz           acq10                   \
 116                                                     \
 117         __asm cmp           dl, 3                   \
 118         __asm sbb           eax, eax                \
 119                                                     \
 120         __asm exit_acq:                             \
 121         __asm mov           Acq, al                 \
 122 }
______unchanged_portion_omitted_

 146 #endif /* __ACWIN_H__ */
```

****
3121 Thu Dec 26 13:50:04 2013
new/usr/src/common/acpica/include/platform/acwin64.h
acpica-unix2-20130823
PANKOVs restructure
****

```
/******************************************************************************
 *
 * Name: acwin64.h - OS specific defines, etc.
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * Copyright (C) 2000 - 2011, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACWIN64_H__
#define __ACWIN64_H__

/*! [Begin] no source code translation (Keep the include) */

#include "acintel.h"
/*! [End] no source code translation !*/

#define ACPI_MACHINE_WIDTH          64

#define ACPI_USE_STANDARD_HEADERS

/*
 * Handle platform- and compiler-specific assembly language differences.
 *
 * Notes:
```



```
 * 1) Interrupt 3 is used to break into a debugger
 * 2) Interrupts are turned off during ACPI register setup
 */

/*! [Begin] no source code translation  */

#define ACPI_FLUSH_CPU_CACHE()

/*
 * For Acpi applications, we don't want to try to access the global lock
 */
#ifdef ACPI_APPLICATION
#define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)       if (AcpiGbl_GlobalLockPresent
#define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd)       if (AcpiGbl_GlobalLockPresent
#else

#define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)

#define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd)

#endif

/*! [End] no source code translation !*/

#endif /* __ACWIN_H__ */
```

```
**********************************************************
   24958 Thu Dec 26 13:50:05 2013
new/usr/src/common/acpica/os_specific/service_layers/osfreebsdtbl.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: osfreebsdtbl - FreeBSD OSL for obtaining ACPI tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpidump.h"

#include <kenv.h>
#include <unistd.h>
#include <sys/param.h>
#include <sys/sysctl.h>


#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("osfreebsdtbl")


/* Local prototypes */

static ACPI_STATUS
OslTableInitialize (
    void);
```



```
static ACPI_STATUS
OslMapTable (
    ACPI_SIZE               Address,
    char                    *Signature,
    ACPI_TABLE_HEADER       **Table);

static ACPI_STATUS
OslAddTablesToList (
    void);

static ACPI_STATUS
OslGetTableViaRoot (
    char                    *Signature,
    UINT32                  Instance,
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address);


/* Hints for RSDP */

#define SYSTEM_KENV         "hint.acpi.0.rsdp"
#define SYSTEM_SYSCTL       "machdep.acpi_root"

/* Initialization flags */

UINT8                   Gbl_TableListInitialized = FALSE;
UINT8                   Gbl_MainTableObtained = FALSE;

/* Local copies of main ACPI tables */

ACPI_TABLE_RSDP         Gbl_Rsdp;
ACPI_TABLE_FADT         *Gbl_Fadt;
ACPI_TABLE_RSDT         *Gbl_Rsdt;
ACPI_TABLE_XSDT         *Gbl_Xsdt;

/* Fadt address */

ACPI_PHYSICAL_ADDRESS   Gbl_FadtAddress;

/* Revision of RSD PTR */

UINT8                   Gbl_Revision;

/* List of information about obtained ACPI tables */

typedef struct          table_info
{
    struct table_info       *Next;
    char                    Signature[4];
    UINT32                  Instance;
    ACPI_PHYSICAL_ADDRESS   Address;

} OSL_TABLE_INFO;

OSL_TABLE_INFO          *Gbl_TableListHead = NULL;


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByAddress
 *
 * PARAMETERS:  Address         - Physical address of the ACPI table
 *              Table           - Where a pointer to the table is returned
 *
 * RETURN:      Status; Table buffer is returned if AE_OK.
 *              AE_NOT_FOUND: A valid table was not found at the address
```

```
 *
 * DESCRIPTION: Get an ACPI table via a physical memory address.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByAddress (
    ACPI_PHYSICAL_ADDRESS   Address,
    ACPI_TABLE_HEADER       **Table)
{
    ACPI_TABLE_HEADER       *MappedTable;
    ACPI_TABLE_HEADER       *LocalTable;
    ACPI_STATUS             Status;


    /* Validate the input physical address to avoid program crash */

    if (Address < ACPI_HI_RSDP_WINDOW_BASE)
    {
        fprintf (stderr, "Invalid table address: 0x%8.8X%8.8X\n",
            ACPI_FORMAT_UINT64 (Address));
        return (AE_BAD_ADDRESS);
    }

    /* Map the table and validate it */

    Status = OslMapTable (Address, NULL, &MappedTable);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Copy table to local buffer and return it */

    LocalTable = calloc (1, MappedTable->Length);
    if (!LocalTable)
    {
        AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
        return (AE_NO_MEMORY);
    }

    ACPI_MEMCPY (LocalTable, MappedTable, MappedTable->Length);
    AcpiOsUnmapMemory (MappedTable, MappedTable->Length);

    *Table = LocalTable;
    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByName
 *
 * PARAMETERS:  Signature       - ACPI Signature for desired table. Must be
 *                                a null terminated 4-character string.
 *              Instance        - Multiple table support for SSDT/UEFI (0...n)
 *                                Must be 0 for other tables.
 *              Table           - Where a pointer to the table is returned
 *              Address         - Where the table physical address is returned
 *
 * RETURN:      Status; Table buffer and physical address returned if AE_OK.
 *              AE_LIMIT: Instance is beyond valid limit
 *              AE_NOT_FOUND: A table with the signature was not found
 *
 * NOTE:        Assumes the input signature is uppercase.
 *
```



```
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByName (
    char                    *Signature,
    UINT32                  Instance,
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address)
{
    ACPI_STATUS             Status;


    /* Instance is only valid for SSDT/UEFI tables */

    if (Instance &&
        !ACPI_COMPARE_NAME (Signature, ACPI_SIG_SSDT) &&
        !ACPI_COMPARE_NAME (Signature, ACPI_SIG_UEFI))
    {
        return (AE_LIMIT);
    }

    /* Initialize main tables */

    Status = OslTableInitialize ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /*
     * If one of the main ACPI tables was requested (RSDT/XSDT/FADT),
     * simply return it immediately.
     */
    if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_XSDT))
    {
        if (!Gbl_Revision)
        {
            return (AE_NOT_FOUND);
        }

        *Address = Gbl_Rsdp.XsdtPhysicalAddress;
        *Table = (ACPI_TABLE_HEADER *) Gbl_Xsdt;
        return (AE_OK);
    }

    if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_RSDT))
    {
        if (!Gbl_Rsdp.RsdtPhysicalAddress)
        {
            return (AE_NOT_FOUND);
        }

        *Address = Gbl_Rsdp.RsdtPhysicalAddress;
        *Table = (ACPI_TABLE_HEADER *) Gbl_Rsdt;
        return (AE_OK);
    }

    if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_FADT))
    {
        *Address = Gbl_FadtAddress;
        *Table = (ACPI_TABLE_HEADER *) Gbl_Fadt;
        return (AE_OK);
    }

    /* Not a main ACPI table, attempt to extract it from the RSDT/XSDT */
```

```
260     Status = OslGetTableViaRoot (Signature, Instance, Table, Address);
261     if (ACPI_FAILURE (Status))
262     {
263         return (Status);
264     }

266     return (AE_OK);
267 }


270 /******************************************************************************
271  *
272  * FUNCTION:    AcpiOsGetTableByIndex
273  *
274  * PARAMETERS:  Index           - Which table to get
275  *              Table           - Where a pointer to the table is returned
276  *              Instance        - Where a pointer to the table instance no. is
277  *                                returned
278  *              Address         - Where the table physical address is returned
279  *
280  * RETURN:      Status; Table buffer and physical address returned if AE_OK.
281  *              AE_LIMIT: Index is beyond valid limit
282  *
283  * DESCRIPTION: Get an ACPI table via an index value (0 through n). Returns
284  *              AE_LIMIT when an invalid index is reached. Index is not
285  *              necessarily an index into the RSDT/XSDT.
286  *
287  *****************************************************************************/

289 ACPI_STATUS
290 AcpiOsGetTableByIndex (
291     UINT32                  Index,
292     ACPI_TABLE_HEADER       **Table,
293     UINT32                  *Instance,
294     ACPI_PHYSICAL_ADDRESS   *Address)
295 {
296     OSL_TABLE_INFO          *Info;
297     ACPI_STATUS             Status;
298     UINT32                  i;


301     /* Initialize main tables */

303     Status = OslTableInitialize ();
304     if (ACPI_FAILURE (Status))
305     {
306         return (Status);
307     }

309     /* Add all tables to list */

311     Status = OslAddTablesToList ();
312     if (ACPI_FAILURE (Status))
313     {
314         return (Status);
315     }

317     /* Validate Index */

319     if (Index >= Gbl_TableListHead->Instance)
320     {
321         return (AE_LIMIT);
322     }

324     /* Point to the table list entry specified by the Index argument */
```



```
326     Info = Gbl_TableListHead;
327     for (i = 0; i <= Index; i++)
328     {
329         Info = Info->Next;
330     }

332     /* Now we can just get the table via the address or name */

334     if (Info->Address)
335     {
336         Status = AcpiOsGetTableByAddress (Info->Address, Table);
337         if (ACPI_SUCCESS (Status))
338         {
339             *Address = Info->Address;
340         }
341     }
342     else
343     {
344         Status = AcpiOsGetTableByName (Info->Signature, Info->Instance,
345             Table, Address);
346     }

348     if (ACPI_SUCCESS (Status))
349     {
350         *Instance = Info->Instance;
351     }
352     return (Status);
353 }


356 /******************************************************************************
357  *
358  * FUNCTION:    OslTableInitialize
359  *
360  * PARAMETERS:  None
361  *
362  * RETURN:      Status
363  *
364  * DESCRIPTION: Initialize ACPI table data. Get and store main ACPI tables to
365  *              local variables. Main ACPI tables include RSDP, FADT, RSDT,
366  *              and/or XSDT.
367  *
368  *****************************************************************************/

370 static ACPI_STATUS
371 OslTableInitialize (
372     void)
373 {
374     char                    Buffer[32];
375     ACPI_TABLE_HEADER       *MappedTable;
376     UINT8                   *TableAddress;
377     UINT8                   *RsdpAddress;
378     ACPI_PHYSICAL_ADDRESS   RsdpBase;
379     ACPI_SIZE               RsdpSize;
380     ACPI_STATUS             Status;
381     u_long                  Address = 0;
382     size_t                  Length = sizeof (Address);


385     /* Get main ACPI tables from memory on first invocation of this function */

387     if (Gbl_MainTableObtained)
388     {
389         return (AE_OK);
390     }
```

```
392     /* Attempt to use kenv or sysctl to find RSD PTR record. */
394     if (Gbl_RsdpBase)
395     {
396         Address = Gbl_RsdpBase;
397     }
398     else if (kenv (KENV_GET, SYSTEM_KENV, Buffer, sizeof (Buffer)) > 0)
399     {
400         Address = ACPI_STRTOUL (Buffer, NULL, 0);
401     }
402     if (!Address)
403     {
404         if (sysctlbyname (SYSTEM_SYSCTL, &Address, &Length, NULL, 0) != 0)
405         {
406             Address = 0;
407         }
408     }
409     if (Address)
410     {
411         RsdpBase = Address;
412         RsdpSize = sizeof (Gbl_Rsdp);
413     }
414     else
415     {
416         RsdpBase = ACPI_HI_RSDP_WINDOW_BASE;
417         RsdpSize = ACPI_HI_RSDP_WINDOW_SIZE;
418     }
420     /* Get RSDP from memory */
422     RsdpAddress = AcpiOsMapMemory (RsdpBase, RsdpSize);
423     if (!RsdpAddress)
424     {
425         return (AE_BAD_ADDRESS);
426     }
428     /* Search low memory for the RSDP */
430     TableAddress = AcpiTbScanMemoryForRsdp (RsdpAddress, RsdpSize);
431     if (!TableAddress)
432     {
433         AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
434         return (AE_ERROR);
435     }
437     ACPI_MEMCPY (&Gbl_Rsdp, TableAddress, sizeof (Gbl_Rsdp));
438     AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
440     /* Get XSDT from memory */
442     if (Gbl_Rsdp.Revision)
443     {
444         Status = OslMapTable (Gbl_Rsdp.XsdtPhysicalAddress,
445             ACPI_SIG_XSDT, &MappedTable);
446         if (ACPI_FAILURE (Status))
447         {
448             return (Status);
449         }
451         Gbl_Revision = 2;
452         Gbl_Xsdt = calloc (1, MappedTable->Length);
453         if (!Gbl_Xsdt)
454         {
455             fprintf (stderr,
456                 "XSDT: Could not allocate buffer for table of length %X\n",
457                 MappedTable->Length);
```



```
458             AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
459             return (AE_NO_MEMORY);
460         }
462         ACPI_MEMCPY (Gbl_Xsdt, MappedTable, MappedTable->Length);
463         AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
464     }
466     /* Get RSDT from memory */
468     if (Gbl_Rsdp.RsdtPhysicalAddress)
469     {
470         Status = OslMapTable (Gbl_Rsdp.RsdtPhysicalAddress,
471             ACPI_SIG_RSDT, &MappedTable);
472         if (ACPI_FAILURE (Status))
473         {
474             return (Status);
475         }
477         Gbl_Rsdt = calloc (1, MappedTable->Length);
478         if (!Gbl_Rsdt)
479         {
480             fprintf (stderr,
481                 "RSDT: Could not allocate buffer for table of length %X\n",
482                 MappedTable->Length);
483             AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
484             return (AE_NO_MEMORY);
485         }
487         ACPI_MEMCPY (Gbl_Rsdt, MappedTable, MappedTable->Length);
488         AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
489     }
491     /* Get FADT from memory */
493     if (Gbl_Revision)
494     {
495         Gbl_FadtAddress = Gbl_Xsdt->TableOffsetEntry[0];
496     }
497     else
498     {
499         Gbl_FadtAddress = Gbl_Rsdt->TableOffsetEntry[0];
500     }
502     if (!Gbl_FadtAddress)
503     {
504         fprintf(stderr, "FADT: Table could not be found\n");
505         return (AE_ERROR);
506     }
508     Status = OslMapTable (Gbl_FadtAddress, ACPI_SIG_FADT, &MappedTable);
509     if (ACPI_FAILURE (Status))
510     {
511         return (Status);
512     }
514     Gbl_Fadt = calloc (1, MappedTable->Length);
515     if (!Gbl_Fadt)
516     {
517         fprintf (stderr,
518             "FADT: Could not allocate buffer for table of length %X\n",
519             MappedTable->Length);
520         AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
521         return (AE_NO_MEMORY);
522     }
```

```
524     ACPI_MEMCPY (Gbl_Fadt, MappedTable, MappedTable->Length);
525     AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
526     Gbl_MainTableObtained = TRUE;
527     return (AE_OK);
528 }


531 /******************************************************************************
532  *
533  * FUNCTION:    OslGetTableViaRoot
534  *
535  * PARAMETERS:  Signature       - ACPI Signature for common table. Must be
536  *                               a null terminated 4-character string.
537  *              Instance        - Multiple table support for SSDT/UEFI (0...n)
538  *                               Must be 0 for other tables.
539  *              Table           - Where a pointer to the table is returned
540  *              Address         - Where the table physical address is returned
541  *
542  * RETURN:      Status; Table buffer and physical address returned if AE_OK.
543  *              AE_LIMIT: Instance is beyond valid limit
544  *              AE_NOT_FOUND: A table with the signature was not found
545  *
546  * DESCRIPTION: Get an ACPI table via the root table (RSDT/XSDT)
547  *
548  * NOTE:        Assumes the input signature is uppercase.
549  *
550  *****************************************************************************/

552 static ACPI_STATUS
553 OslGetTableViaRoot (
554     char                    *Signature,
555     UINT32                  Instance,
556     ACPI_TABLE_HEADER       **Table,
557     ACPI_PHYSICAL_ADDRESS   *Address)
558 {
559     ACPI_TABLE_HEADER       *LocalTable = NULL;
560     ACPI_TABLE_HEADER       *MappedTable = NULL;
561     UINT8                   NumberOfTables;
562     UINT32                  CurrentInstance = 0;
563     ACPI_PHYSICAL_ADDRESS   TableAddress = 0;
564     ACPI_STATUS             Status;
565     UINT32                  i;


568     /* DSDT and FACS address must be extracted from the FADT */

570     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT) ||
571         ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS))
572     {
573         /*
574          * Get the appropriate address, either 32-bit or 64-bit. Be very
575          * careful about the FADT length and validate table addresses.
576          * Note: The 64-bit addresses have priority.
577          */
578         if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT))
579         {
580             if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_XDSDT) &&
581                 Gbl_Fadt->XDsdt)
582             {
583                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->XDsdt;
584             }
585             else if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_DSDT) &&
586                 Gbl_Fadt->Dsdt)
587             {
588                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->Dsdt;
589             }
```



```
590         }
591         else /* FACS */
592         {
593             if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_XFACS) &&
594                 Gbl_Fadt->XFacs)
595             {
596                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->XFacs;
597             }
598             else if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_FACS) &&
599                 Gbl_Fadt->Facs)
600             {
601                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->Facs;
602             }
603         }
604     }
605     else /* Case for a normal ACPI table */
606     {
607         if (Gbl_Revision)
608         {
609             NumberOfTables =
610                 (Gbl_Xsdt->Header.Length - sizeof (Gbl_Xsdt->Header))
611                 / sizeof (Gbl_Xsdt->TableOffsetEntry[0]);
612         }
613         else /* Use RSDT if XSDT is not available */
614         {
615             NumberOfTables =
616                 (Gbl_Rsdt->Header.Length - sizeof (Gbl_Rsdt->Header))
617                 / sizeof (Gbl_Rsdt->TableOffsetEntry[0]);
618         }

620         /* Search RSDT/XSDT for the requested table */

622         for (i = 0; i < NumberOfTables; i++)
623         {
624             if (Gbl_Revision)
625             {
626                 TableAddress = Gbl_Xsdt->TableOffsetEntry[i];
627             }
628             else
629             {
630                 TableAddress = Gbl_Rsdt->TableOffsetEntry[i];
631             }

633             MappedTable = AcpiOsMapMemory (TableAddress, sizeof (*MappedTable));
634             if (!MappedTable)
635             {
636                 return (AE_BAD_ADDRESS);
637             }

639             /* Does this table match the requested signature? */

641             if (ACPI_COMPARE_NAME (MappedTable->Signature, Signature))
642             {

644                 /* Match table instance (for SSDT/UEFI tables) */

646                 if (CurrentInstance == Instance)
647                 {
648                     AcpiOsUnmapMemory (MappedTable, sizeof (*MappedTable));
649                     break;
650                 }

652                 CurrentInstance++;
653             }

655             AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
```

```
656             TableAddress = 0;
657         }
658     }

660     if (!TableAddress)
661     {
662         if (CurrentInstance)
663         {
664             return (AE_LIMIT);
665         }
666         return (AE_NOT_FOUND);
667     }

669     /* Now we can get the requested table */

671     Status = OslMapTable (TableAddress, Signature, &MappedTable);
672     if (ACPI_FAILURE (Status))
673     {
674         return (Status);
675     }

677     /* Copy table to local buffer and return it */

679     LocalTable = calloc (1, MappedTable->Length);
680     if (!LocalTable)
681     {
682         AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
683         return (AE_NO_MEMORY);
684     }

686     ACPI_MEMCPY (LocalTable, MappedTable, MappedTable->Length);
687     AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
688     *Table = LocalTable;
689     *Address = TableAddress;
690     return (AE_OK);
691 }


694 /*******************************************************************************
695  *
696  * FUNCTION:    OslAddTablesToList
697  *
698  * PARAMETERS:  None
699  *
700  * RETURN:      Status; Table list is initialized if AE_OK.
701  *
702  * DESCRIPTION: Add ACPI tables to the table list.
703  *
704  ******************************************************************************/

706 static ACPI_STATUS
707 OslAddTablesToList(
708     void)
709 {
710     ACPI_PHYSICAL_ADDRESS   TableAddress;
711     OSL_TABLE_INFO          *Info = NULL;
712     OSL_TABLE_INFO          *NewInfo;
713     ACPI_TABLE_HEADER       *Table;
714     UINT8                   Instance;
715     UINT8                   NumberOfTables;
716     int                     i;


719     /* Initialize the table list on first invocation */

721     if (Gbl_TableListInitialized)
```



```
722     {
723         return (AE_OK);
724     }

726     /* Add mandatory tables to global table list first */

728     for (i = 0; i < 4; i++)
729     {
730         NewInfo = calloc (1, sizeof (*NewInfo));
731         if (!NewInfo)
732         {
733             return (AE_NO_MEMORY);
734         }

736         switch (i) {
737         case 0:

739             Gbl_TableListHead = Info = NewInfo;
740             continue;

742         case 1:

744             ACPI_MOVE_NAME (NewInfo->Signature,
745                 Gbl_Revision ? ACPI_SIG_XSDT : ACPI_SIG_RSDT);
746             break;

748         case 2:

750             ACPI_MOVE_NAME (NewInfo->Signature, ACPI_SIG_FACS);
751             break;

753         default:

755             ACPI_MOVE_NAME (NewInfo->Signature, ACPI_SIG_DSDT);

757         }

759         Info->Next = NewInfo;
760         Info = NewInfo;
761         Gbl_TableListHead->Instance++;
762     }

764     /* Add normal tables from RSDT/XSDT to global list */

766     if (Gbl_Revision)
767     {
768         NumberOfTables =
769             (Gbl_Xsdt->Header.Length - sizeof (Gbl_Xsdt->Header))
770             / sizeof (Gbl_Xsdt->TableOffsetEntry[0]);
771     }
772     else
773     {
774         NumberOfTables =
775             (Gbl_Rsdt->Header.Length - sizeof (Gbl_Rsdt->Header))
776             / sizeof (Gbl_Rsdt->TableOffsetEntry[0]);
777     }

779     for (i = 0; i < NumberOfTables; i++)
780     {
781         if (Gbl_Revision)
782         {
783             TableAddress = Gbl_Xsdt->TableOffsetEntry[i];
784         }
785         else
786         {
787             TableAddress = Gbl_Rsdt->TableOffsetEntry[i];
```

```
788         }

790         Table = AcpiOsMapMemory (TableAddress, sizeof (*Table));
791         if (!Table)
792         {
793             return (AE_BAD_ADDRESS);
794         }

796         Instance = 0;
797         NewInfo = Gbl_TableListHead;
798         while (NewInfo->Next != NULL)
799         {
800             NewInfo = NewInfo->Next;
801             if (ACPI_COMPARE_NAME (Table->Signature, NewInfo->Signature))
802             {
803                 Instance++;
804             }
805         }

807         NewInfo = calloc (1, sizeof (*NewInfo));
808         if (!NewInfo)
809         {
810             AcpiOsUnmapMemory (Table, sizeof (*Table));
811             return (AE_NO_MEMORY);
812         }

814         ACPI_MOVE_NAME (NewInfo->Signature, Table->Signature);

816         AcpiOsUnmapMemory (Table, sizeof (*Table));

818         NewInfo->Instance = Instance;
819         NewInfo->Address = TableAddress;
820         Info->Next = NewInfo;
821         Info = NewInfo;
822         Gbl_TableListHead->Instance++;
823     }

825     Gbl_TableListInitialized = TRUE;
826     return (AE_OK);
827 }


830 /*******************************************************************************
831  *
832  * FUNCTION:    OslMapTable
833  *
834  * PARAMETERS:  Address             - Address of the table in memory
835  *              Signature           - Optional ACPI Signature for desired table.
836  *                                    Null terminated 4-character string.
837  *              Table               - Where a pointer to the mapped table is
838  *                                    returned
839  *
840  * RETURN:      Status; Mapped table is returned if AE_OK.
841  *
842  * DESCRIPTION: Map entire ACPI table into caller's address space. Also
843  *              validates the table and checksum.
844  *
845  ******************************************************************************/

847 static ACPI_STATUS
848 OslMapTable (
849     ACPI_SIZE               Address,
850     char                    *Signature,
851     ACPI_TABLE_HEADER       **Table)
852 {
853     ACPI_TABLE_HEADER       *MappedTable;
```



```
854     UINT32                  Length;


857     /* Map the header so we can get the table length */

859     MappedTable = AcpiOsMapMemory (Address, sizeof (*MappedTable));
860     if (!MappedTable)
861     {
862         return (AE_BAD_ADDRESS);
863     }

865     /* Check if table is valid */

867     if (!ApIsValidHeader (MappedTable))
868     {
869         AcpiOsUnmapMemory (MappedTable, sizeof (*MappedTable));
870         return (AE_BAD_HEADER);
871     }

873     /* If specified, signature must match */

875     if (Signature &&
876         !ACPI_COMPARE_NAME (Signature, MappedTable->Signature))
877     {
878         AcpiOsUnmapMemory (MappedTable, sizeof (*MappedTable));
879         return (AE_NOT_EXIST);
880     }

882     /* Map the entire table */

884     Length = MappedTable->Length;
885     AcpiOsUnmapMemory (MappedTable, sizeof (*MappedTable));

887     MappedTable = AcpiOsMapMemory (Address, Length);
888     if (!MappedTable)
889     {
890         return (AE_BAD_ADDRESS);
891     }

893     (void) ApIsValidChecksum (MappedTable);

895     *Table = MappedTable;

897     return (AE_OK);
898 }
```

```
**********************************************************
   39197 Thu Dec 26 13:50:05 2013
new/usr/src/common/acpica/os_specific/service_layers/oslinuxtbl.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: oslinuxtbl - Linux OSL for obtaining ACPI tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpidump.h"


#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("oslinuxtbl")


#ifndef PATH_MAX
#define PATH_MAX 256
#endif


/* List of information about obtained ACPI tables */

typedef struct          table_info
{
    struct table_info       *Next;
    UINT32                  Instance;
    char                    Signature[ACPI_NAME_SIZE];
} OSL_TABLE_INFO;

/* Local prototypes */

static ACPI_STATUS
OslTableInitialize (
    void);

static ACPI_STATUS
OslTableNameFromFile (
    char                    *Filename,
    char                    *Signature,
    UINT32                  *Instance);

static ACPI_STATUS
OslAddTableToList (
    char                    *Signature,
    UINT32                  Instance);

static ACPI_STATUS
OslReadTableFromFile (
    char                    *Filename,
    ACPI_SIZE               FileOffset,
    char                    *Signature,
    ACPI_TABLE_HEADER       **Table);

static ACPI_STATUS
OslMapTable (
    ACPI_SIZE               Address,
    char                    *Signature,
    ACPI_TABLE_HEADER       **Table);

static void
OslUnmapTable (
    ACPI_TABLE_HEADER       *Table);

static ACPI_PHYSICAL_ADDRESS
OslFindRsdpViaEfi (
    void);

static ACPI_STATUS
OslLoadRsdp (
    void);

static ACPI_STATUS
OslListCustomizedTables (
    char                    *Directory);

static ACPI_STATUS
OslGetCustomizedTable (
    char                    *Pathname,
    char                    *Signature,
    UINT32                  Instance,
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address);

static ACPI_STATUS
OslListBiosTables (
    void);

static ACPI_STATUS
OslGetBiosTable (
    char                    *Signature,
    UINT32                  Instance,
```

```
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address);

static ACPI_STATUS
OslGetLastStatus (
    ACPI_STATUS             DefaultStatus);


/* File locations */

#define DYNAMIC_TABLE_DIR   "/sys/firmware/acpi/tables/dynamic"
#define STATIC_TABLE_DIR    "/sys/firmware/acpi/tables"
#define EFI_SYSTAB          "/sys/firmware/efi/systab"

/* Should we get dynamically loaded SSDTs from DYNAMIC_TABLE_DIR? */

UINT8                   Gbl_DumpDynamicTables = TRUE;

/* Initialization flags */

UINT8                   Gbl_TableListInitialized = FALSE;

/* Local copies of main ACPI tables */

ACPI_TABLE_RSDP         Gbl_Rsdp;
ACPI_TABLE_FADT         *Gbl_Fadt = NULL;
ACPI_TABLE_RSDT         *Gbl_Rsdt = NULL;
ACPI_TABLE_XSDT         *Gbl_Xsdt = NULL;

/* Table addresses */

ACPI_PHYSICAL_ADDRESS   Gbl_FadtAddress = 0;
ACPI_PHYSICAL_ADDRESS   Gbl_RsdpAddress = 0;

/* Revision of RSD PTR */

UINT8                   Gbl_Revision = 0;

OSL_TABLE_INFO          *Gbl_TableListHead = NULL;
UINT32                  Gbl_TableCount = 0;


/******************************************************************************
 *
 * FUNCTION:    OslGetLastStatus
 *
 * PARAMETERS:  DefaultStatus   - Default error status to return
 *
 * RETURN:      Status; Converted from errno.
 *
 * DESCRIPTION: Get last errno and conver it to ACPI_STATUS.
 *
 *****************************************************************************/

static ACPI_STATUS
OslGetLastStatus (
    ACPI_STATUS             DefaultStatus)
{

    switch (errno)
    {
    case EACCES:
    case EPERM:

        return (AE_ACCESS);
```



```
    case ENOENT:

        return (AE_NOT_FOUND);

    case ENOMEM:

        return (AE_NO_MEMORY);

    default:

        return (DefaultStatus);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByAddress
 *
 * PARAMETERS:  Address         - Physical address of the ACPI table
 *              Table           - Where a pointer to the table is returned
 *
 * RETURN:      Status; Table buffer is returned if AE_OK.
 *              AE_NOT_FOUND: A valid table was not found at the address
 *
 * DESCRIPTION: Get an ACPI table via a physical memory address.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByAddress (
    ACPI_PHYSICAL_ADDRESS   Address,
    ACPI_TABLE_HEADER       **Table)
{
    UINT32                  TableLength;
    ACPI_TABLE_HEADER       *MappedTable;
    ACPI_TABLE_HEADER       *LocalTable = NULL;
    ACPI_STATUS             Status = AE_OK;


    /* Get main ACPI tables from memory on first invocation of this function */

    Status = OslTableInitialize ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Map the table and validate it */

    Status = OslMapTable (Address, NULL, &MappedTable);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Copy table to local buffer and return it */

    TableLength = ApGetTableLength (MappedTable);
    if (TableLength == 0)
    {
        Status = AE_BAD_HEADER;
        goto ErrorExit;
    }

    LocalTable = calloc (1, TableLength);
```

```
260     if (!LocalTable)
261     {
262         Status = AE_NO_MEMORY;
263         goto ErrorExit;
264     }

266     ACPI_MEMCPY (LocalTable, MappedTable, TableLength);

268 ErrorExit:
269     OslUnmapTable (MappedTable);
270     *Table = LocalTable;
271     return (AE_OK);
272 }


275 /******************************************************************************
276  *
277  * FUNCTION:    AcpiOsGetTableByName
278  *
279  * PARAMETERS:  Signature       - ACPI Signature for desired table. Must be
280  *                               a null terminated 4-character string.
281  *              Instance        - Multiple table support for SSDT/UEFI (0...n)
282  *                               Must be 0 for other tables.
283  *              Table           - Where a pointer to the table is returned
284  *              Address         - Where the table physical address is returned
285  *
286  * RETURN:      Status; Table buffer and physical address returned if AE_OK.
287  *              AE_LIMIT: Instance is beyond valid limit
288  *              AE_NOT_FOUND: A table with the signature was not found
289  *
290  * NOTE:        Assumes the input signature is uppercase.
291  *
292  *****************************************************************************/

294 ACPI_STATUS
295 AcpiOsGetTableByName (
296     char                    *Signature,
297     UINT32                  Instance,
298     ACPI_TABLE_HEADER       **Table,
299     ACPI_PHYSICAL_ADDRESS   *Address)
300 {
301     ACPI_STATUS             Status;


304     /* Get main ACPI tables from memory on first invocation of this function */

306     Status = OslTableInitialize ();
307     if (ACPI_FAILURE (Status))
308     {
309         return (Status);
310     }

312     /* Not a main ACPI table, attempt to extract it from the RSDT/XSDT */

314     if (!Gbl_DumpCustomizedTables)
315     {
316         /* Attempt to get the table from the memory */

318         Status = OslGetBiosTable (Signature, Instance, Table, Address);
319     }
320     else
321     {
322         /* Attempt to get the table from the static directory */

324         Status = OslGetCustomizedTable (STATIC_TABLE_DIR, Signature,
325             Instance, Table, Address);
```



```
326     }

328     if (ACPI_FAILURE (Status) && Status == AE_LIMIT)
329     {
330         if (Gbl_DumpDynamicTables)
331         {
332             /* Attempt to get a dynamic table */

334             Status = OslGetCustomizedTable (DYNAMIC_TABLE_DIR, Signature,
335                 Instance, Table, Address);
336         }
337     }

339     return (Status);
340 }


343 /******************************************************************************
344  *
345  * FUNCTION:    OslAddTableToList
346  *
347  * PARAMETERS:  Signature       - Table signature
348  *              Instance        - Table instance
349  *
350  * RETURN:      Status; Successfully added if AE_OK.
351  *              AE_NO_MEMORY: Memory allocation error
352  *
353  * DESCRIPTION: Insert a table structure into OSL table list.
354  *
355  *****************************************************************************/

357 static ACPI_STATUS
358 OslAddTableToList (
359     char                    *Signature,
360     UINT32                  Instance)
361 {
362     OSL_TABLE_INFO          *NewInfo;
363     OSL_TABLE_INFO          *Next;
364     UINT32                  NextInstance = 0;
365     BOOLEAN                 Found = FALSE;


368     NewInfo = calloc (1, sizeof (OSL_TABLE_INFO));
369     if (!NewInfo)
370     {
371         return (AE_NO_MEMORY);
372     }

374     ACPI_MOVE_NAME (NewInfo->Signature, Signature);

376     if (!Gbl_TableListHead)
377     {
378         Gbl_TableListHead = NewInfo;
379     }
380     else
381     {
382         Next = Gbl_TableListHead;
383         while (1)
384         {
385             if (ACPI_COMPARE_NAME (Next->Signature, Signature))
386             {
387                 if (Next->Instance == Instance)
388                 {
389                     Found = TRUE;
390                 }
391                 if (Next->Instance >= NextInstance)
```

```
392                 {
393                     NextInstance = Next->Instance + 1;
394                 }
395             }

397             if (!Next->Next)
398             {
399                 break;
400             }
401             Next = Next->Next;
402         }
403         Next->Next = NewInfo;
404     }

406     if (Found)
407     {
408         if (Instance)
409         {
410             fprintf (stderr,
411                 "%4.4s: Warning unmatched table instance %d, expected %d\n",
412                 Signature, Instance, NextInstance);
413         }
414         Instance = NextInstance;
415     }

417     NewInfo->Instance = Instance;
418     Gbl_TableCount++;

420     return (AE_OK);
421 }


424 /******************************************************************************
425  *
426  * FUNCTION:    AcpiOsGetTableByIndex
427  *
428  * PARAMETERS:  Index           - Which table to get
429  *              Table           - Where a pointer to the table is returned
430  *              Instance        - Where a pointer to the table instance no. is
431  *                                returned
432  *              Address         - Where the table physical address is returned
433  *
434  * RETURN:      Status; Table buffer and physical address returned if AE_OK.
435  *              AE_LIMIT: Index is beyond valid limit
436  *
437  * DESCRIPTION: Get an ACPI table via an index value (0 through n). Returns
438  *              AE_LIMIT when an invalid index is reached. Index is not
439  *              necessarily an index into the RSDT/XSDT.
440  *
441  *****************************************************************************/

443 ACPI_STATUS
444 AcpiOsGetTableByIndex (
445     UINT32                  Index,
446     ACPI_TABLE_HEADER       **Table,
447     UINT32                  *Instance,
448     ACPI_PHYSICAL_ADDRESS   *Address)
449 {
450     OSL_TABLE_INFO          *Info;
451     ACPI_STATUS             Status;
452     UINT32                  i;


455     /* Get main ACPI tables from memory on first invocation of this function */

457     Status = OslTableInitialize ();
```



```
458     if (ACPI_FAILURE (Status))
459     {
460         return (Status);
461     }

463     /* Validate Index */

465     if (Index >= Gbl_TableCount)
466     {
467         return (AE_LIMIT);
468     }

470     /* Point to the table list entry specified by the Index argument */

472     Info = Gbl_TableListHead;
473     for (i = 0; i < Index; i++)
474     {
475         Info = Info->Next;
476     }

478     /* Now we can just get the table via the signature */

480     Status = AcpiOsGetTableByName (Info->Signature, Info->Instance,
481         Table, Address);

483     if (ACPI_SUCCESS (Status))
484     {
485         *Instance = Info->Instance;
486     }
487     return (Status);
488 }


491 /******************************************************************************
492  *
493  * FUNCTION:    OslFindRsdpViaEfi
494  *
495  * PARAMETERS:  None
496  *
497  * RETURN:      RSDP address if found
498  *
499  * DESCRIPTION: Find RSDP address via EFI.
500  *
501  *****************************************************************************/

503 static ACPI_PHYSICAL_ADDRESS
504 OslFindRsdpViaEfi (
505     void)
506 {
507     FILE                    *File;
508     char                    Buffer[80];
509     unsigned long           Address = 0;


512     File = fopen (EFI_SYSTAB, "r");
513     if (File)
514     {
515         while (fgets (Buffer, 80, File))
516         {
517             if (sscanf (Buffer, "ACPI20=0x%lx", &Address) == 1)
518             {
519                 break;
520             }
521         }
522         fclose (File);
523     }
```

```
525     return ((ACPI_PHYSICAL_ADDRESS) (Address));
526 }


529 /*****************************************************************************
530  *
531  * FUNCTION:    OslLoadRsdp
532  *
533  * PARAMETERS:  None
534  *
535  * RETURN:      Status
536  *
537  * DESCRIPTION: Scan and load RSDP.
538  *
539  *****************************************************************************/

541 static ACPI_STATUS
542 OslLoadRsdp (
543     void)
544 {
545     ACPI_TABLE_HEADER       *MappedTable;
546     UINT8                   *RsdpAddress;
547     ACPI_PHYSICAL_ADDRESS   RsdpBase;
548     ACPI_SIZE               RsdpSize;


551     /* Get RSDP from memory */

553     RsdpSize = sizeof (ACPI_TABLE_RSDP);
554     if (Gbl_RsdpBase)
555     {
556         RsdpBase = Gbl_RsdpBase;
557     }
558     else
559     {
560         RsdpBase = OslFindRsdpViaEfi ();
561     }

563     if (!RsdpBase)
564     {
565         RsdpBase = ACPI_HI_RSDP_WINDOW_BASE;
566         RsdpSize = ACPI_HI_RSDP_WINDOW_SIZE;
567     }

569     RsdpAddress = AcpiOsMapMemory (RsdpBase, RsdpSize);
570     if (!RsdpAddress)
571     {
572         return (OslGetLastStatus (AE_BAD_ADDRESS));
573     }

575     /* Search low memory for the RSDP */

577     MappedTable = ACPI_CAST_PTR (ACPI_TABLE_HEADER,
578         AcpiTbScanMemoryForRsdp (RsdpAddress, RsdpSize));
579     if (!MappedTable)
580     {
581         AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
582         return (AE_NOT_FOUND);
583     }

585     Gbl_RsdpAddress = RsdpBase + (ACPI_CAST8 (MappedTable) - RsdpAddress);

587     ACPI_MEMCPY (&Gbl_Rsdp, MappedTable, sizeof (ACPI_TABLE_RSDP));
588     AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
```



```
590     return (AE_OK);
591 }


594 /*****************************************************************************
595  *
596  * FUNCTION:    OslTableInitialize
597  *
598  * PARAMETERS:  None
599  *
600  * RETURN:      Status
601  *
602  * DESCRIPTION: Initialize ACPI table data. Get and store main ACPI tables to
603  *              local variables. Main ACPI tables include RSDT, FADT, RSDT,
604  *              and/or XSDT.
605  *
606  *****************************************************************************/

608 static ACPI_STATUS
609 OslTableInitialize (
610     void)
611 {
612     ACPI_STATUS             Status;
613     ACPI_PHYSICAL_ADDRESS   Address;


616     if (Gbl_TableListInitialized)
617     {
618         return (AE_OK);
619     }

621     /* Get RSDP from memory */

623     Status = OslLoadRsdp ();
624     if (ACPI_FAILURE (Status))
625     {
626         return (Status);
627     }

629     /* Get XSDT from memory */

631     if (Gbl_Rsdp.Revision)
632     {
633         if (Gbl_Xsdt)
634         {
635             free (Gbl_Xsdt);
636             Gbl_Xsdt = NULL;
637         }

639         Gbl_Revision = 2;
640         Status = OslGetBiosTable (ACPI_SIG_XSDT, 0,
641             ACPI_CAST_PTR (ACPI_TABLE_HEADER *, &Gbl_Xsdt), &Address);
642         if (ACPI_FAILURE (Status))
643         {
644             return (Status);
645         }
646     }

648     /* Get RSDT from memory */

650     if (Gbl_Rsdp.RsdtPhysicalAddress)
651     {
652         if (Gbl_Rsdt)
653         {
654             free (Gbl_Rsdt);
655             Gbl_Rsdt = NULL;
```

```
656         }

658         Status = OslGetBiosTable (ACPI_SIG_RSDT, 0,
659             ACPI_CAST_PTR (ACPI_TABLE_HEADER *, &Gbl_Rsdt), &Address);
660         if (ACPI_FAILURE (Status))
661         {
662             return (Status);
663         }
664     }

666     /* Get FADT from memory */

668     if (Gbl_Fadt)
669     {
670         free (Gbl_Fadt);
671         Gbl_Fadt = NULL;
672     }

674     Status = OslGetBiosTable (ACPI_SIG_FADT, 0,
675         ACPI_CAST_PTR (ACPI_TABLE_HEADER *, &Gbl_Fadt), &Gbl_FadtAddress);
676     if (ACPI_FAILURE (Status))
677     {
678         return (Status);
679     }

681     if (!Gbl_DumpCustomizedTables)
682     {
683         /* Add mandatory tables to global table list first */

685         Status = OslAddTableToList (AP_DUMP_SIG_RSDP, 0);
686         if (ACPI_FAILURE (Status))
687         {
688             return (Status);
689         }

691         Status = OslAddTableToList (ACPI_SIG_RSDT, 0);
692         if (ACPI_FAILURE (Status))
693         {
694             return (Status);
695         }

697         if (Gbl_Revision == 2)
698         {
699             Status = OslAddTableToList (ACPI_SIG_XSDT, 0);
700             if (ACPI_FAILURE (Status))
701             {
702                 return (Status);
703             }
704         }

706         Status = OslAddTableToList (ACPI_SIG_DSDT, 0);
707         if (ACPI_FAILURE (Status))
708         {
709             return (Status);
710         }

712         Status = OslAddTableToList (ACPI_SIG_FACS, 0);
713         if (ACPI_FAILURE (Status))
714         {
715             return (Status);
716         }

718         /* Add all tables found in the memory */

720         Status = OslListBiosTables ();
721         if (ACPI_FAILURE (Status))
```



```
722         {
723             return (Status);
724         }
725     }
726     else
727     {
728         /* Add all tables found in the static directory */

730         Status = OslListCustomizedTables (STATIC_TABLE_DIR);
731         if (ACPI_FAILURE (Status))
732         {
733             return (Status);
734         }
735     }

737     if (Gbl_DumpDynamicTables)
738     {
739         /* Add all dynamically loaded tables in the dynamic directory */

741         Status = OslListCustomizedTables (DYNAMIC_TABLE_DIR);
742         if (ACPI_FAILURE (Status))
743         {
744             return (Status);
745         }
746     }

748     Gbl_TableListInitialized = TRUE;
749     return (AE_OK);
750 }


753 /******************************************************************************
754  *
755  * FUNCTION:    OslListBiosTables
756  *
757  * PARAMETERS:  None
758  *
759  * RETURN:      Status; Table list is initialized if AE_OK.
760  *
761  * DESCRIPTION: Add ACPI tables to the table list from memory.
762  *
763  * NOTE:        This works on Linux as table customization does not modify the
764  *              addresses stored in RSDP/RSDT/XSDT/FADT.
765  *
766  *****************************************************************************/

768 static ACPI_STATUS
769 OslListBiosTables (
770     void)
771 {
772     ACPI_TABLE_HEADER       *MappedTable = NULL;
773     UINT8                   *TableData;
774     UINT8                   NumberOfTables;
775     UINT8                   ItemSize;
776     ACPI_PHYSICAL_ADDRESS   TableAddress = 0;
777     ACPI_STATUS             Status = AE_OK;
778     UINT32                  i;


781     if (Gbl_Revision)
782     {
783         ItemSize = sizeof (UINT64);
784         TableData = ACPI_CAST8 (Gbl_Xsdt) + sizeof (ACPI_TABLE_HEADER);
785         NumberOfTables =
786             (UINT8) ((Gbl_Xsdt->Header.Length - sizeof (ACPI_TABLE_HEADER))
787             / ItemSize);
```

```
788     }
789     else /* Use RSDT if XSDT is not available */
790     {
791         ItemSize = sizeof (UINT32);
792         TableData = ACPI_CAST8 (Gbl_Rsdt) + sizeof (ACPI_TABLE_HEADER);
793         NumberOfTables =
794             (UINT8) ((Gbl_Rsdt->Header.Length - sizeof (ACPI_TABLE_HEADER))
795             / ItemSize);
796     }

798     /* Search RSDT/XSDT for the requested table */

800     for (i = 0; i < NumberOfTables; ++i, TableData += ItemSize)
801     {
802         if (Gbl_Revision)
803         {
804             TableAddress =
805                 (ACPI_PHYSICAL_ADDRESS) (*ACPI_CAST64 (TableData));
806         }
807         else
808         {
809             TableAddress =
810                 (ACPI_PHYSICAL_ADDRESS) (*ACPI_CAST32 (TableData));
811         }

813         Status = OslMapTable (TableAddress, NULL, &MappedTable);
814         if (ACPI_FAILURE (Status))
815         {
816             return (Status);
817         }

819         OslAddTableToList (MappedTable->Signature, 0);
820         OslUnmapTable (MappedTable);
821     }

823     return (AE_OK);
824 }


827 /******************************************************************************
828  *
829  * FUNCTION:    OslGetBiosTable
830  *
831  * PARAMETERS:  Signature       - ACPI Signature for common table. Must be
832  *                                a null terminated 4-character string.
833  *              Instance        - Multiple table support for SSDT/UEFI (0...n)
834  *                                Must be 0 for other tables.
835  *              Table           - Where a pointer to the table is returned
836  *              Address         - Where the table physical address is returned
837  *
838  * RETURN:      Status; Table buffer and physical address returned if AE_OK.
839  *              AE_LIMIT: Instance is beyond valid limit
840  *              AE_NOT_FOUND: A table with the signature was not found
841  *
842  * DESCRIPTION: Get a BIOS provided ACPI table
843  *
844  * NOTE:        Assumes the input signature is uppercase.
845  *
846  *****************************************************************************/

848 static ACPI_STATUS
849 OslGetBiosTable (
850     char                    *Signature,
851     UINT32                  Instance,
852     ACPI_TABLE_HEADER       **Table,
853     ACPI_PHYSICAL_ADDRESS   *Address)
```



```
854 {
855     ACPI_TABLE_HEADER       *LocalTable = NULL;
856     ACPI_TABLE_HEADER       *MappedTable = NULL;
857     UINT8                   *TableData;
858     UINT8                   NumberOfTables;
859     UINT8                   ItemSize;
860     UINT32                  CurrentInstance = 0;
861     ACPI_PHYSICAL_ADDRESS   TableAddress = 0;
862     UINT32                  TableLength = 0;
863     ACPI_STATUS             Status = AE_OK;
864     UINT32                  i;


867     /* Handle special tables whose addresses are not in RSDT/XSDT */

869     if (ACPI_COMPARE_NAME (Signature, AP_DUMP_SIG_RSDP) ||
870         ACPI_COMPARE_NAME (Signature, ACPI_SIG_RSDT) ||
871         ACPI_COMPARE_NAME (Signature, ACPI_SIG_XSDT) ||
872         ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT) ||
873         ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS))
874     {
875         /*
876          * Get the appropriate address, either 32-bit or 64-bit. Be very
877          * careful about the FADT length and validate table addresses.
878          * Note: The 64-bit addresses have priority.
879          */
880         if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT))
881         {
882             if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_XDSDT) &&
883                 Gbl_Fadt->XDsdt)
884             {
885                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->XDsdt;
886             }
887             else if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_DSDT) &&
888                 Gbl_Fadt->Dsdt)
889             {
890                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->Dsdt;
891             }
892         }
893         else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS))
894         {
895             if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_XFACS) &&
896                 Gbl_Fadt->XFacs)
897             {
898                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->XFacs;
899             }
900             else if ((Gbl_Fadt->Header.Length >= MIN_FADT_FOR_FACS) &&
901                 Gbl_Fadt->Facs)
902             {
903                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->Facs;
904             }
905         }
906         else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_XSDT))
907         {
908             if (!Gbl_Revision)
909             {
910                 return (AE_BAD_SIGNATURE);
911             }
912             TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Rsdp.XsdtPhysicalAddress;
913         }
914         else if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_RSDT))
915         {
916             TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Rsdp.RsdtPhysicalAddress;
917         }
918         else
919         {
```

```
920             TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_RsdpAddress;
921             Signature = ACPI_SIG_RSDP;
922         }

924         /* Now we can get the requested special table */

926         Status = OslMapTable (TableAddress, Signature, &MappedTable);
927         if (ACPI_FAILURE (Status))
928         {
929             return (Status);
930         }

932         TableLength = ApGetTableLength (MappedTable);
933     }
934     else /* Case for a normal ACPI table */
935     {
936         if (Gbl_Revision)
937         {
938             ItemSize = sizeof (UINT64);
939             TableData = ACPI_CAST8 (Gbl_Xsdt) + sizeof (ACPI_TABLE_HEADER);
940             NumberOfTables =
941                 (UINT8) ((Gbl_Xsdt->Header.Length - sizeof (ACPI_TABLE_HEADER))
942                 / ItemSize);
943         }
944         else /* Use RSDT if XSDT is not available */
945         {
946             ItemSize = sizeof (UINT32);
947             TableData = ACPI_CAST8 (Gbl_Rsdt) + sizeof (ACPI_TABLE_HEADER);
948             NumberOfTables =
949                 (UINT8) ((Gbl_Rsdt->Header.Length - sizeof (ACPI_TABLE_HEADER))
950                 / ItemSize);
951         }

953         /* Search RSDT/XSDT for the requested table */

955         for (i = 0; i < NumberOfTables; ++i, TableData += ItemSize)
956         {
957             if (Gbl_Revision)
958             {
959                 TableAddress =
960                     (ACPI_PHYSICAL_ADDRESS) (*ACPI_CAST64 (TableData));
961             }
962             else
963             {
964                 TableAddress =
965                     (ACPI_PHYSICAL_ADDRESS) (*ACPI_CAST32 (TableData));
966             }

968             Status = OslMapTable (TableAddress, NULL, &MappedTable);
969             if (ACPI_FAILURE (Status))
970             {
971                 return (Status);
972             }
973             TableLength = MappedTable->Length;

975             /* Does this table match the requested signature? */

977             if (!ACPI_COMPARE_NAME (MappedTable->Signature, Signature))
978             {
979                 OslUnmapTable (MappedTable);
980                 MappedTable = NULL;
981                 continue;
982             }

984             /* Match table instance (for SSDT/UEFI tables) */
```



```
986             if (CurrentInstance != Instance)
987             {
988                 OslUnmapTable (MappedTable);
989                 MappedTable = NULL;
990                 CurrentInstance++;
991                 continue;
992             }

994             break;
995         }
996     }

998     if (!MappedTable)
999     {
1000        return (AE_LIMIT);
1001    }

1003    if (TableLength == 0)
1004    {
1005        Status = AE_BAD_HEADER;
1006        goto ErrorExit;
1007    }

1009    /* Copy table to local buffer and return it */

1011    LocalTable = calloc (1, TableLength);
1012    if (!LocalTable)
1013    {
1014        Status = AE_NO_MEMORY;
1015        goto ErrorExit;
1016    }

1018    ACPI_MEMCPY (LocalTable, MappedTable, TableLength);
1019    *Address = TableAddress;
1020    *Table = LocalTable;

1022 ErrorExit:
1023    OslUnmapTable (MappedTable);
1024    return (AE_OK);
1025 }


1028 /******************************************************************************
1029  *
1030  * FUNCTION:    OslListCustomizedTables
1031  *
1032  * PARAMETERS:  Directory           - Directory that contains the tables
1033  *
1034  * RETURN:      Status; Table list is initialized if AE_OK.
1035  *
1036  * DESCRIPTION: Add ACPI tables to the table list from a directory.
1037  *
1038  *****************************************************************************/

1040 static ACPI_STATUS
1041 OslListCustomizedTables (
1042     char                    *Directory)
1043 {
1044     void                    *TableDir;
1045     UINT32                  Instance;
1046     char                    TempName[ACPI_NAME_SIZE];
1047     char                    *Filename;
1048     ACPI_STATUS             Status = AE_OK;


1051    /* Open the requested directory */
```

```
1053     TableDir = AcpiOsOpenDirectory (Directory, "*", REQUEST_FILE_ONLY);
1054     if (!TableDir)
1055     {
1056         return (OslGetLastStatus (AE_NOT_FOUND));
1057     }

1059     /* Examine all entries in this directory */

1061     while ((Filename = AcpiOsGetNextFilename (TableDir)))
1062     {
1063         /* Extract table name and instance number */

1065         Status = OslTableNameFromFile (Filename, TempName, &Instance);

1067         /* Ignore meaningless files */

1069         if (ACPI_FAILURE (Status))
1070         {
1071             continue;
1072         }

1074         /* Add new info node to global table list */

1076         Status = OslAddTableToList (TempName, Instance);
1077         if (ACPI_FAILURE (Status))
1078         {
1079             break;
1080         }
1081     }

1083     AcpiOsCloseDirectory (TableDir);
1084     return (Status);
1085 }


1088 /******************************************************************************
1089  *
1090  * FUNCTION:    OslMapTable
1091  *
1092  * PARAMETERS:  Address             - Address of the table in memory
1093  *              Signature           - Optional ACPI Signature for desired table.
1094  *                                    Null terminated 4-character string.
1095  *              Table               - Where a pointer to the mapped table is
1096  *                                    returned
1097  *
1098  * RETURN:      Status; Mapped table is returned if AE_OK.
1099  *              AE_NOT_FOUND: A valid table was not found at the address
1100  *
1101  * DESCRIPTION: Map entire ACPI table into caller's address space.
1102  *
1103  *****************************************************************************/

1105 static ACPI_STATUS
1106 OslMapTable (
1107     ACPI_SIZE               Address,
1108     char                    *Signature,
1109     ACPI_TABLE_HEADER       **Table)
1110 {
1111     ACPI_TABLE_HEADER       *MappedTable;
1112     UINT32                  Length;


1115     if (!Address)
1116     {
1117         return (AE_BAD_ADDRESS);
```



```
1118     }

1120     /*
1121      * Map the header so we can get the table length.
1122      * Use sizeof (ACPI_TABLE_HEADER) as:
1123      * 1. it is bigger than 24 to include RSDP->Length
1124      * 2. it is smaller than sizeof (ACPI_TABLE_RSDP)
1125      */
1126     MappedTable = AcpiOsMapMemory (Address, sizeof (ACPI_TABLE_HEADER));
1127     if (!MappedTable)
1128     {
1129         fprintf (stderr, "Could not map table header at 0x%8.8X%8.8X\n",
1130             ACPI_FORMAT_UINT64 (Address));
1131         return (OslGetLastStatus (AE_BAD_ADDRESS));
1132     }

1134     /* If specified, signature must match */

1136     if (Signature &&
1137         !ACPI_COMPARE_NAME (Signature, MappedTable->Signature))
1138     {
1139         AcpiOsUnmapMemory (MappedTable, sizeof (ACPI_TABLE_HEADER));
1140         return (AE_BAD_SIGNATURE);
1141     }

1143     /* Map the entire table */

1145     Length = ApGetTableLength (MappedTable);
1146     AcpiOsUnmapMemory (MappedTable, sizeof (ACPI_TABLE_HEADER));
1147     if (Length == 0)
1148     {
1149         return (AE_BAD_HEADER);
1150     }

1152     MappedTable = AcpiOsMapMemory (Address, Length);
1153     if (!MappedTable)
1154     {
1155         fprintf (stderr, "Could not map table at 0x%8.8X%8.8X length %8.8X\n",
1156             ACPI_FORMAT_UINT64 (Address), Length);
1157         return (OslGetLastStatus (AE_INVALID_TABLE_LENGTH));
1158     }

1160     (void) ApIsValidChecksum (MappedTable);

1162     *Table = MappedTable;
1163     return (AE_OK);
1164 }


1167 /******************************************************************************
1168  *
1169  * FUNCTION:    OslUnmapTable
1170  *
1171  * PARAMETERS:  Table               - A pointer to the mapped table
1172  *
1173  * RETURN:      None
1174  *
1175  * DESCRIPTION: Unmap entire ACPI table.
1176  *
1177  *****************************************************************************/

1179 static void
1180 OslUnmapTable (
1181     ACPI_TABLE_HEADER       *Table)
1182 {
1183     if (Table)
```

```
    {
        AcpiOsUnmapMemory (Table, ApGetTableLength (Table));
    }
}


/******************************************************************************
 *
 * FUNCTION:    OslTableNameFromFile
 *
 * PARAMETERS:  Filename            - File that contains the desired table
 *              Signature           - Pointer to 4-character buffer to store
 *                                    extracted table signature.
 *              Instance            - Pointer to integer to store extracted
 *                                    table instance number.
 *
 * RETURN:      Status; Table name is extracted if AE_OK.
 *
 * DESCRIPTION: Extract table signature and instance number from a table file
 *              name.
 *
 *****************************************************************************/

static ACPI_STATUS
OslTableNameFromFile (
    char                    *Filename,
    char                    *Signature,
    UINT32                  *Instance)
{

    /* Ignore meaningless files */

    if (strlen (Filename) < ACPI_NAME_SIZE)
    {
        return (AE_BAD_SIGNATURE);
    }

    /* Extract instance number */

    if (isdigit ((int) Filename[ACPI_NAME_SIZE]))
    {
        sscanf (&Filename[ACPI_NAME_SIZE], "%d", Instance);
    }
    else if (strlen (Filename) != ACPI_NAME_SIZE)
    {
        return (AE_BAD_SIGNATURE);
    }
    else
    {
        *Instance = 0;
    }

    /* Extract signature */

    ACPI_MOVE_NAME (Signature, Filename);
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    OslReadTableFromFile
 *
 * PARAMETERS:  Filename            - File that contains the desired table
 *              FileOffset          - Offset of the table in file
 *              Signature           - Optional ACPI Signature for desired table.
```



```
 *                                    A null terminated 4-character string.
 *              Table               - Where a pointer to the table is returned
 *
 * RETURN:      Status; Table buffer is returned if AE_OK.
 *
 * DESCRIPTION: Read a ACPI table from a file.
 *
 *****************************************************************************/

static ACPI_STATUS
OslReadTableFromFile (
    char                    *Filename,
    ACPI_SIZE               FileOffset,
    char                    *Signature,
    ACPI_TABLE_HEADER       **Table)
{
    FILE                    *TableFile;
    ACPI_TABLE_HEADER       Header;
    ACPI_TABLE_HEADER       *LocalTable = NULL;
    UINT32                  TableLength;
    INT32                   Count;
    UINT32                  Total = 0;
    ACPI_STATUS             Status = AE_OK;


    /* Open the file */

    TableFile = fopen (Filename, "rb");
    if (TableFile == NULL)
    {
        fprintf (stderr, "Could not open table file: %s\n", Filename);
        return (OslGetLastStatus (AE_NOT_FOUND));
    }

    fseek (TableFile, FileOffset, SEEK_SET);

    /* Read the Table header to get the table length */

    Count = fread (&Header, 1, sizeof (ACPI_TABLE_HEADER), TableFile);
    if (Count != sizeof (ACPI_TABLE_HEADER))
    {
        fprintf (stderr, "Could not read table header: %s\n", Filename);
        Status = AE_BAD_HEADER;
        goto ErrorExit;
    }

    /* If signature is specified, it must match the table */

    if (Signature &&
        !ACPI_COMPARE_NAME (Signature, Header.Signature))
    {
        fprintf (stderr, "Incorrect signature: Expecting %4.4s, found %4.4s\n",
            Signature, Header.Signature);
        Status = AE_BAD_SIGNATURE;
        goto ErrorExit;
    }

    TableLength = ApGetTableLength (&Header);
    if (TableLength == 0)
    {
        Status = AE_BAD_HEADER;
        goto ErrorExit;
    }

    /* Read the entire table into a local buffer */
```

```
    LocalTable = calloc (1, TableLength);
    if (!LocalTable)
    {
        fprintf (stderr,
            "%4.4s: Could not allocate buffer for table of length %X\n",
            Header.Signature, TableLength);
        Status = AE_NO_MEMORY;
        goto ErrorExit;
    }

    fseek (TableFile, FileOffset, SEEK_SET);

    while (!feof (TableFile) && Total < TableLength)
    {
        Count = fread (LocalTable, 1, TableLength-Total, TableFile);
        if (Count < 0)
        {
            fprintf (stderr, "%4.4s: Could not read table content\n",
                Header.Signature);
            Status = AE_INVALID_TABLE_LENGTH;
            goto ErrorExit;
        }

        Total += Count;
    }

    /* Validate checksum */

    (void) ApIsValidChecksum (LocalTable);

ErrorExit:
    fclose (TableFile);
    *Table = LocalTable;
    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    OslGetCustomizedTable
 *
 * PARAMETERS:  Pathname        - Directory to find Linux customized table
 *              Signature       - ACPI Signature for desired table. Must be
 *                                a null terminated 4-character string.
 *              Instance        - Multiple table support for SSDT/UEFI (0...n)
 *                                Must be 0 for other tables.
 *              Table           - Where a pointer to the table is returned
 *              Address         - Where the table physical address is returned
 *
 * RETURN:      Status; Table buffer is returned if AE_OK.
 *              AE_LIMIT: Instance is beyond valid limit
 *              AE_NOT_FOUND: A table with the signature was not found
 *
 * DESCRIPTION: Get an OS customized table.
 *
 *****************************************************************************/

static ACPI_STATUS
OslGetCustomizedTable (
    char                    *Pathname,
    char                    *Signature,
    UINT32                  Instance,
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address)
{
    void                    *TableDir;
```



```
    UINT32                  CurrentInstance = 0;
    char                    TempName[ACPI_NAME_SIZE];
    char                    TableFilename[PATH_MAX];
    char                    *Filename;
    ACPI_STATUS             Status;


    /* Open the directory for customized tables */

    TableDir = AcpiOsOpenDirectory (Pathname, "*", REQUEST_FILE_ONLY);
    if (!TableDir)
    {
        return (OslGetLastStatus (AE_NOT_FOUND));
    }

    /* Attempt to find the table in the directory */

    while ((Filename = AcpiOsGetNextFilename (TableDir)))
    {
        /* Ignore meaningless files */

        if (!ACPI_COMPARE_NAME (Filename, Signature))
        {
            continue;
        }

        /* Extract table name and instance number */

        Status = OslTableNameFromFile (Filename, TempName, &CurrentInstance);

        /* Ignore meaningless files */

        if (ACPI_FAILURE (Status) || CurrentInstance != Instance)
        {
            continue;
        }

        /* Create the table pathname */

        if (Instance != 0)
        {
            sprintf (TableFilename, "%s/%4.4s%d", Pathname, TempName, Instance);
        }
        else
        {
            sprintf (TableFilename, "%s/%4.4s", Pathname, TempName);
        }
        break;
    }

    AcpiOsCloseDirectory (TableDir);

    if (!Filename)
    {
        return (AE_LIMIT);
    }

    /* There is no physical address saved for customized tables, use zero */

    *Address = 0;
    Status = OslReadTableFromFile (TableFilename, 0, NULL, Table);

    return (Status);
}
```

```
**********************************************************
    7065 Thu Dec 26 13:50:05 2013
new/usr/src/common/acpica/os_specific/service_layers/osunixdir.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: osunixdir - Unix directory access interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include <acpi.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <fnmatch.h>
#include <ctype.h>
#include <sys/stat.h>

/*
 * Allocated structure returned from OsOpenDirectory
 */
typedef struct ExternalFindInfo
{
    char                        *DirPathname;
```



```
    DIR                         *DirPtr;
    char                        temp_buffer[256];
    char                        *WildcardSpec;
    char                        RequestedFileType;

} EXTERNAL_FIND_INFO;


/******************************************************************************
 *
 * FUNCTION:    AcpiOsOpenDirectory
 *
 * PARAMETERS:  DirPathname         - Full pathname to the directory
 *              WildcardSpec        - string of the form "*.c", etc.
 *
 * RETURN:      A directory "handle" to be used in subsequent search operations.
 *              NULL returned on failure.
 *
 * DESCRIPTION: Open a directory in preparation for a wildcard search
 *
 *****************************************************************************/

void *
AcpiOsOpenDirectory (
    char                    *DirPathname,
    char                    *WildcardSpec,
    char                    RequestedFileType)
{
    EXTERNAL_FIND_INFO      *ExternalInfo;
    DIR                     *dir;


    /* Allocate the info struct that will be returned to the caller */

    ExternalInfo = calloc (1, sizeof (EXTERNAL_FIND_INFO));
    if (!ExternalInfo)
    {
        return (NULL);
    }

    /* Get the directory stream */

    dir = opendir (DirPathname);
    if (!dir)
    {
        fprintf (stderr, "Cannot open directory - %s\n", DirPathname);
        free (ExternalInfo);
        return (NULL);
    }

    /* Save the info in the return structure */

    ExternalInfo->WildcardSpec = WildcardSpec;
    ExternalInfo->RequestedFileType = RequestedFileType;
    ExternalInfo->DirPathname = DirPathname;
    ExternalInfo->DirPtr = dir;
    return (ExternalInfo);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetNextFilename
 *
 * PARAMETERS:  DirHandle           - Created via AcpiOsOpenDirectory
 *
```

```
 * RETURN:      Next filename matched. NULL if no more matches.
 *
 * DESCRIPTION: Get the next file in the directory that matches the wildcard
 *              specification.
 *
 *****************************************************************************/

char *
AcpiOsGetNextFilename (
    void                    *DirHandle)
{
    EXTERNAL_FIND_INFO      *ExternalInfo = DirHandle;
    struct dirent           *dir_entry;
    char                    *temp_str;
    int                     str_len;
    struct stat             temp_stat;
    int                     err;


    while ((dir_entry = readdir (ExternalInfo->DirPtr)))
    {
        if (!fnmatch (ExternalInfo->WildcardSpec, dir_entry->d_name, 0))
        {
            if (dir_entry->d_name[0] == '.')
            {
                continue;
            }

            str_len = strlen (dir_entry->d_name) +
                        strlen (ExternalInfo->DirPathname) + 2;

            temp_str = calloc (str_len, 1);
            if (!temp_str)
            {
                fprintf (stderr,
                    "Could not allocate buffer for temporary string\n");
                return (NULL);
            }

            strcpy (temp_str, ExternalInfo->DirPathname);
            strcat (temp_str, "/");
            strcat (temp_str, dir_entry->d_name);

            err = stat (temp_str, &temp_stat);
            if (err == -1)
            {
                fprintf (stderr,
                    "Cannot stat file (should not happen) - %s\n",
                    temp_str);
                free (temp_str);
                return (NULL);
            }

            free (temp_str);

            if ((S_ISDIR (temp_stat.st_mode)
                && (ExternalInfo->RequestedFileType == REQUEST_DIR_ONLY))
               ||
               ((!S_ISDIR (temp_stat.st_mode)
                && ExternalInfo->RequestedFileType == REQUEST_FILE_ONLY)))
            {
                /* copy to a temp buffer because dir_entry struct is on the stac

                strcpy (ExternalInfo->temp_buffer, dir_entry->d_name);
                return (ExternalInfo->temp_buffer);
            }
```



```
        }
    }

    return (NULL);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsCloseDirectory
 *
 * PARAMETERS:  DirHandle           - Created via AcpiOsOpenDirectory
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Close the open directory and cleanup.
 *
 *****************************************************************************/

void
AcpiOsCloseDirectory (
    void                    *DirHandle)
{
    EXTERNAL_FIND_INFO      *ExternalInfo = DirHandle;


    /* Close the directory and free allocations */

    closedir (ExternalInfo->DirPtr);
    free (DirHandle);
}
```

```
**********************************************************
    5179 Thu Dec 26 13:50:05 2013
new/usr/src/common/acpica/os_specific/service_layers/osunixmap.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: osunixmap - Unix OSL for file mappings
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpidump.h"
#include <unistd.h>
#include <sys/mman.h>
#ifdef _FreeBSD
#include <sys/param.h>
#endif

#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("osunixmap")


#ifndef O_BINARY
#define O_BINARY 0
#endif

#ifdef _FreeBSD
#define MMAP_FLAGS          MAP_SHARED
#else
```



```
#define MMAP_FLAGS          MAP_PRIVATE
#endif

#define SYSTEM_MEMORY       "/dev/mem"


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetPageSize
 *
 * PARAMETERS:  None
 *
 * RETURN:      Page size of the platform.
 *
 * DESCRIPTION: Obtain page size of the platform.
 *
 *****************************************************************************/

static ACPI_SIZE
AcpiOsGetPageSize (
    void)
{

#ifdef PAGE_SIZE
    return PAGE_SIZE;
#else
    return sysconf (_SC_PAGESIZE);
#endif
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsMapMemory
 *
 * PARAMETERS:  Where               - Physical address of memory to be mapped
 *              Length              - How much memory to map
 *
 * RETURN:      Pointer to mapped memory. Null on error.
 *
 * DESCRIPTION: Map physical memory into local address space.
 *
 *****************************************************************************/

void *
AcpiOsMapMemory (
    ACPI_PHYSICAL_ADDRESS   Where,
    ACPI_SIZE               Length)
{
    UINT8                   *MappedMemory;
    ACPI_PHYSICAL_ADDRESS   Offset;
    ACPI_SIZE               PageSize;
    int                     fd;


    fd = open (SYSTEM_MEMORY, O_RDONLY | O_BINARY);
    if (fd < 0)
    {
        fprintf (stderr, "Cannot open %s\n", SYSTEM_MEMORY);
        return (NULL);
    }

    /* Align the offset to use mmap */

    PageSize = AcpiOsGetPageSize ();
    Offset = Where % PageSize;
```

```
129     /* Map the table header to get the length of the full table */

131     MappedMemory = mmap (NULL, (Length + Offset), PROT_READ, MMAP_FLAGS,
132         fd, (Where - Offset));
133     if (MappedMemory == MAP_FAILED)
134     {
135         fprintf (stderr, "Cannot map %s\n", SYSTEM_MEMORY);
136         close (fd);
137         return (NULL);
138     }

140     close (fd);
141     return (ACPI_CAST8 (MappedMemory + Offset));
142 }


145 /******************************************************************************
146  *
147  * FUNCTION:    AcpiOsUnmapMemory
148  *
149  * PARAMETERS:  Where               - Logical address of memory to be unmapped
150  *              Length              - How much memory to unmap
151  *
152  * RETURN:      None.
153  *
154  * DESCRIPTION: Delete a previously created mapping. Where and Length must
155  *              correspond to a previous mapping exactly.
156  *
157  *****************************************************************************/

159 void
160 AcpiOsUnmapMemory (
161     void                    *Where,
162     ACPI_SIZE               Length)
163 {
164     ACPI_PHYSICAL_ADDRESS   Offset;
165     ACPI_SIZE               PageSize;


168     PageSize = AcpiOsGetPageSize ();
169     Offset = (ACPI_PHYSICAL_ADDRESS) Where % PageSize;
170     munmap ((UINT8 *) Where - Offset, (Length + Offset));
171 }
```

```
**********************************************************
   35731 Thu Dec 26 13:50:06 2013
new/usr/src/common/acpica/os_specific/service_layers/osunixxf.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: osunixxf - UNIX OSL interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


/*
 * These interfaces are required in order to compile the ASL compiler and the
 * various ACPICA tools under Linux or other Unix-like system.
 */
#include "acpi.h"
#include "accommon.h"
#include "amlcode.h"
#include "acparser.h"
#include "acdebug.h"

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
```



```
#include <unistd.h>
#include <sys/time.h>
#include <semaphore.h>
#include <pthread.h>
#include <errno.h>

#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("osunixxf")


FILE                            *AcpiGbl_OutputFile;
BOOLEAN                         AcpiGbl_DebugTimeout = FALSE;


/* Upcalls to AcpiExec */

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);

void
AeTableOverride (
    ACPI_TABLE_HEADER       *ExistingTable,
    ACPI_TABLE_HEADER       **NewTable);

typedef void* (*PTHREAD_CALLBACK) (void *);

/* Buffer used by AcpiOsVprintf */

#define ACPI_VPRINTF_BUFFER_SIZE    512
#define _ASCII_NEWLINE              '\n'

/* Terminal support for AcpiExec only */

#ifdef ACPI_EXEC_APP
#include <termios.h>

struct termios              OriginalTermAttributes;

ACPI_STATUS
AcpiUtReadLine (
    char                    *Buffer,
    UINT32                  BufferLength,
    UINT32                  *BytesRead);

static void
OsEnterLineEditMode (
    void);

static void
OsExitLineEditMode (
    void);


/******************************************************************************
 *
 * FUNCTION:    OsEnterLineEditMode, OsExitLineEditMode
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Enter/Exit the raw character input mode for the terminal.
 *
 * Interactive line-editing support for the AML debugger. Used with the
 * common/acgetline module.
```

```
 *
 * readline() is not used because of non-portability. It is not available
 * on all systems, and if it is, often the package must be manually installed.
 *
 * Therefore, we use the POSIX tcgetattr/tcsetattr and do the minimal line
 * editing that we need in AcpiOsGetLine.
 *
 * If the POSIX tcgetattr/tcsetattr interfaces are unavailable, these
 * calls will also work:
 *     For OsEnterLineEditMode: system ("stty cbreak -echo")
 *     For OsExitLineEditMode:  system ("stty cooked echo")
 *
 *****************************************************************************/

static void
OsEnterLineEditMode (
    void)
{
    struct termios          LocalTermAttributes;


    /* Get and keep the original attributes */

    if (tcgetattr (STDIN_FILENO, &OriginalTermAttributes))
    {
        fprintf (stderr, "Could not get/set terminal attributes!\n");
        return;
    }

    /* Set the new attributes to enable raw character input */

    memcpy (&LocalTermAttributes, &OriginalTermAttributes,
        sizeof (struct termios));

    LocalTermAttributes.c_lflag &= ~(ICANON | ECHO);
    LocalTermAttributes.c_cc[VMIN] = 1;
    LocalTermAttributes.c_cc[VTIME] = 0;

    tcsetattr (STDIN_FILENO, TCSANOW, &LocalTermAttributes);
}

static void
OsExitLineEditMode (
    void)
{
    /* Set terminal attributes back to the original values */

    tcsetattr (STDIN_FILENO, TCSANOW, &OriginalTermAttributes);
}


#else

/* These functions are not needed for other ACPICA utilities */

#define OsEnterLineEditMode()
#define OsExitLineEditMode()
#endif


/******************************************************************************
 *
 * FUNCTION:    AcpiOsInitialize, AcpiOsTerminate
 *
 * PARAMETERS:  None
 *
```



```
 * RETURN:      Status
 *
 * DESCRIPTION: Initialize and terminate this module.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsInitialize (
    void)
{

    AcpiGbl_OutputFile = stdout;

    OsEnterLineEditMode ();
    return (AE_OK);
}

ACPI_STATUS
AcpiOsTerminate (
    void)
{

    OsExitLineEditMode ();
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetRootPointer
 *
 * PARAMETERS:  None
 *
 * RETURN:      RSDP physical address
 *
 * DESCRIPTION: Gets the ACPI root pointer (RSDP)
 *
 *****************************************************************************/

ACPI_PHYSICAL_ADDRESS
AcpiOsGetRootPointer (
    void)
{

    return (AeLocalGetRootPointer ());
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsPredefinedOverride
 *
 * PARAMETERS:  InitVal             - Initial value of the predefined object
 *              NewVal              - The new value for the object
 *
 * RETURN:      Status, pointer to value. Null pointer returned if not
 *              overriding.
 *
 * DESCRIPTION: Allow the OS to override predefined names
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsPredefinedOverride (
    const ACPI_PREDEFINED_NAMES *InitVal,
    ACPI_STRING                 *NewVal)
```

```
256 {
258     if (!InitVal || !NewVal)
259     {
260         return (AE_BAD_PARAMETER);
261     }
263     *NewVal = NULL;
264     return (AE_OK);
265 }
268 /******************************************************************************
269  *
270  * FUNCTION:    AcpiOsTableOverride
271  *
272  * PARAMETERS:  ExistingTable       - Header of current table (probably
273  *                                    firmware)
274  *              NewTable            - Where an entire new table is returned.
275  *
276  * RETURN:      Status, pointer to new table. Null pointer returned if no
277  *              table is available to override
278  *
279  * DESCRIPTION: Return a different version of a table if one is available
280  *
281  *****************************************************************************/
283 ACPI_STATUS
284 AcpiOsTableOverride (
285     ACPI_TABLE_HEADER       *ExistingTable,
286     ACPI_TABLE_HEADER       **NewTable)
287 {
289     if (!ExistingTable || !NewTable)
290     {
291         return (AE_BAD_PARAMETER);
292     }
294     *NewTable = NULL;
296 #ifdef ACPI_EXEC_APP
298     AeTableOverride (ExistingTable, NewTable);
299     return (AE_OK);
300 #else
302     return (AE_NO_ACPI_TABLES);
303 #endif
304 }
307 /******************************************************************************
308  *
309  * FUNCTION:    AcpiOsPhysicalTableOverride
310  *
311  * PARAMETERS:  ExistingTable       - Header of current table (probably firmware
312  *              NewAddress          - Where new table address is returned
313  *                                    (Physical address)
314  *              NewTableLength      - Where new table length is returned
315  *
316  * RETURN:      Status, address/length of new table. Null pointer returned
317  *              if no table is available to override.
318  *
319  * DESCRIPTION: Returns AE_SUPPORT, function not used in user space.
320  *
321  *****************************************************************************/
```



```
323 ACPI_STATUS
324 AcpiOsPhysicalTableOverride (
325     ACPI_TABLE_HEADER       *ExistingTable,
326     ACPI_PHYSICAL_ADDRESS   *NewAddress,
327     UINT32                  *NewTableLength)
328 {
330     return (AE_SUPPORT);
331 }
334 /******************************************************************************
335  *
336  * FUNCTION:    AcpiOsRedirectOutput
337  *
338  * PARAMETERS:  Destination         - An open file handle/pointer
339  *
340  * RETURN:      None
341  *
342  * DESCRIPTION: Causes redirect of AcpiOsPrintf and AcpiOsVprintf
343  *
344  *****************************************************************************/
346 void
347 AcpiOsRedirectOutput (
348     void                    *Destination)
349 {
351     AcpiGbl_OutputFile = Destination;
352 }
355 /******************************************************************************
356  *
357  * FUNCTION:    AcpiOsPrintf
358  *
359  * PARAMETERS:  fmt, ...            - Standard printf format
360  *
361  * RETURN:      None
362  *
363  * DESCRIPTION: Formatted output. Note: very similar to AcpiOsVprintf
364  *              (performance), changes should be tracked in both functions.
365  *
366  *****************************************************************************/
368 void ACPI_INTERNAL_VAR_XFACE
369 AcpiOsPrintf (
370     const char              *Fmt,
371     ...)
372 {
373     va_list                 Args;
374     UINT8                   Flags;
377     Flags = AcpiGbl_DbOutputFlags;
378     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
379     {
380         /* Output is directable to either a file (if open) or the console */
382         if (AcpiGbl_DebugFile)
383         {
384             /* Output file is open, send the output there */
386             va_start (Args, Fmt);
387             vfprintf (AcpiGbl_DebugFile, Fmt, Args);
```

```
388             va_end (Args);
389         }
390         else
391         {
392             /* No redirection, send output to console (once only!) */

394             Flags |= ACPI_DB_CONSOLE_OUTPUT;
395         }
396     }

398     if (Flags & ACPI_DB_CONSOLE_OUTPUT)
399     {
400         va_start (Args, Fmt);
401         vfprintf (AcpiGbl_OutputFile, Fmt, Args);
402         va_end (Args);
403     }
404 }


407 /******************************************************************************
408  *
409  * FUNCTION:    AcpiOsVprintf
410  *
411  * PARAMETERS:  fmt                 - Standard printf format
412  *              args                - Argument list
413  *
414  * RETURN:      None
415  *
416  * DESCRIPTION: Formatted output with argument list pointer. Note: very
417  *              similar to AcpiOsPrintf, changes should be tracked in both
418  *              functions.
419  *
420  *****************************************************************************/

422 void
423 AcpiOsVprintf (
424     const char              *Fmt,
425     va_list                 Args)
426 {
427     UINT8                   Flags;
428     char                    Buffer[ACPI_VPRINTF_BUFFER_SIZE];


431     /*
432      * We build the output string in a local buffer because we may be
433      * outputting the buffer twice. Using vfprintf is problematic because
434      * some implementations modify the args pointer/structure during
435      * execution. Thus, we use the local buffer for portability.
436      *
437      * Note: Since this module is intended for use by the various ACPICA
438      * utilities/applications, we can safely declare the buffer on the stack.
439      * Also, This function is used for relatively small error messages only.
440      */
441     vsnprintf (Buffer, ACPI_VPRINTF_BUFFER_SIZE, Fmt, Args);

443     Flags = AcpiGbl_DbOutputFlags;
444     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
445     {
446         /* Output is directable to either a file (if open) or the console */

448         if (AcpiGbl_DebugFile)
449         {
450             /* Output file is open, send the output there */

452             fputs (Buffer, AcpiGbl_DebugFile);
453         }
```



```
454         else
455         {
456             /* No redirection, send output to console (once only!) */

458             Flags |= ACPI_DB_CONSOLE_OUTPUT;
459         }
460     }

462     if (Flags & ACPI_DB_CONSOLE_OUTPUT)
463     {
464         fputs (Buffer, AcpiGbl_OutputFile);
465     }
466 }


469 #ifndef ACPI_EXEC_APP
470 /******************************************************************************
471  *
472  * FUNCTION:    AcpiOsGetLine
473  *
474  * PARAMETERS:  Buffer              - Where to return the command line
475  *              BufferLength        - Maximum length of Buffer
476  *              BytesRead           - Where the actual byte count is returned
477  *
478  * RETURN:      Status and actual bytes read
479  *
480  * DESCRIPTION: Get the next input line from the terminal. NOTE: For the
481  *              AcpiExec utility, we use the acgetline module instead to
482  *              provide line-editing and history support.
483  *
484  *****************************************************************************/

486 ACPI_STATUS
487 AcpiOsGetLine (
488     char                    *Buffer,
489     UINT32                  BufferLength,
490     UINT32                  *BytesRead)
491 {
492     int                     InputChar;
493     UINT32                  EndOfLine;


496     /* Standard AcpiOsGetLine for all utilities except AcpiExec */

498     for (EndOfLine = 0; ; EndOfLine++)
499     {
500         if (EndOfLine >= BufferLength)
501         {
502             return (AE_BUFFER_OVERFLOW);
503         }

505         if ((InputChar = getchar ()) == EOF)
506         {
507             return (AE_ERROR);
508         }

510         if (!InputChar || InputChar == _ASCII_NEWLINE)
511         {
512             break;
513         }

515         Buffer[EndOfLine] = (char) InputChar;
516     }

518     /* Null terminate the buffer */
```

```
520     Buffer[EndOfLine] = 0;

522     /* Return the number of bytes in the string */

524     if (BytesRead)
525     {
526         *BytesRead = EndOfLine;
527     }

529     return (AE_OK);
530 }
531 #endif


534 /******************************************************************************
535  *
536  * FUNCTION:    AcpiOsMapMemory
537  *
538  * PARAMETERS:  where               - Physical address of memory to be mapped
539  *              length              - How much memory to map
540  *
541  * RETURN:      Pointer to mapped memory. Null on error.
542  *
543  * DESCRIPTION: Map physical memory into caller's address space
544  *
545  *****************************************************************************/

547 void *
548 AcpiOsMapMemory (
549     ACPI_PHYSICAL_ADDRESS   where,
550     ACPI_SIZE               length)
551 {

553     return (ACPI_TO_POINTER ((ACPI_SIZE) where));
554 }


557 /******************************************************************************
558  *
559  * FUNCTION:    AcpiOsUnmapMemory
560  *
561  * PARAMETERS:  where               - Logical address of memory to be unmapped
562  *              length              - How much memory to unmap
563  *
564  * RETURN:      None.
565  *
566  * DESCRIPTION: Delete a previously created mapping. Where and Length must
567  *              correspond to a previous mapping exactly.
568  *
569  *****************************************************************************/

571 void
572 AcpiOsUnmapMemory (
573     void                    *where,
574     ACPI_SIZE               length)
575 {

577     return;
578 }


581 /******************************************************************************
582  *
583  * FUNCTION:    AcpiOsAllocate
584  *
585  * PARAMETERS:  Size                - Amount to allocate, in bytes
```



```
586  *
587  * RETURN:      Pointer to the new allocation. Null on error.
588  *
589  * DESCRIPTION: Allocate memory. Algorithm is dependent on the OS.
590  *
591  *****************************************************************************/

593 void *
594 AcpiOsAllocate (
595     ACPI_SIZE               size)
596 {
597     void                    *Mem;


600     Mem = (void *) malloc ((size_t) size);
601     return (Mem);
602 }


605 /******************************************************************************
606  *
607  * FUNCTION:    AcpiOsFree
608  *
609  * PARAMETERS:  mem                 - Pointer to previously allocated memory
610  *
611  * RETURN:      None.
612  *
613  * DESCRIPTION: Free memory allocated via AcpiOsAllocate
614  *
615  *****************************************************************************/

617 void
618 AcpiOsFree (
619     void                    *mem)
620 {

622     free (mem);
623 }


626 #ifdef ACPI_SINGLE_THREADED
627 /******************************************************************************
628  *
629  * FUNCTION:    Semaphore stub functions
630  *
631  * DESCRIPTION: Stub functions used for single-thread applications that do
632  *              not require semaphore synchronization. Full implementations
633  *              of these functions appear after the stubs.
634  *
635  *****************************************************************************/

637 ACPI_STATUS
638 AcpiOsCreateSemaphore (
639     UINT32              MaxUnits,
640     UINT32              InitialUnits,
641     ACPI_HANDLE         *OutHandle)
642 {
643     *OutHandle = (ACPI_HANDLE) 1;
644     return (AE_OK);
645 }

647 ACPI_STATUS
648 AcpiOsDeleteSemaphore (
649     ACPI_HANDLE         Handle)
650 {
651     return (AE_OK);
```

```
}

ACPI_STATUS
AcpiOsWaitSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units,
    UINT16              Timeout)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiOsSignalSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units)
{
    return (AE_OK);
}

#else
/******************************************************************************
 *
 * FUNCTION:    AcpiOsCreateSemaphore
 *
 * PARAMETERS:  InitialUnits        - Units to be assigned to the new semaphore
 *              OutHandle           - Where a handle will be returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create an OS semaphore
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsCreateSemaphore (
    UINT32              MaxUnits,
    UINT32              InitialUnits,
    ACPI_HANDLE         *OutHandle)
{
    sem_t               *Sem;


    if (!OutHandle)
    {
        return (AE_BAD_PARAMETER);
    }

#ifdef __APPLE__
    {
        char            *SemaphoreName = tmpnam (NULL);

        Sem = sem_open (SemaphoreName, O_EXCL|O_CREAT, 0755, InitialUnits);
        if (!Sem)
        {
            return (AE_NO_MEMORY);
        }
        sem_unlink (SemaphoreName); /* This just deletes the name */
    }

#else
    Sem = AcpiOsAllocate (sizeof (sem_t));
    if (!Sem)
    {
        return (AE_NO_MEMORY);
    }
```



```
    if (sem_init (Sem, 0, InitialUnits) == -1)
    {
        AcpiOsFree (Sem);
        return (AE_BAD_PARAMETER);
    }
#endif

    *OutHandle = (ACPI_HANDLE) Sem;
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsDeleteSemaphore
 *
 * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Delete an OS semaphore
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsDeleteSemaphore (
    ACPI_HANDLE         Handle)
{
    sem_t               *Sem = (sem_t *) Handle;


    if (!Sem)
    {
        return (AE_BAD_PARAMETER);
    }

    if (sem_destroy (Sem) == -1)
    {
        return (AE_BAD_PARAMETER);
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsWaitSemaphore
 *
 * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
 *              Units               - How many units to wait for
 *              MsecTimeout         - How long to wait (milliseconds)
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Wait for units
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWaitSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units,
    UINT16              MsecTimeout)
{
    ACPI_STATUS         Status = AE_OK;
```

```
    sem_t               *Sem = (sem_t *) Handle;
#ifndef ACPI_USE_ALTERNATE_TIMEOUT
    struct timespec     Time;
    int                 RetVal;
#endif


    if (!Sem)
    {
        return (AE_BAD_PARAMETER);
    }

    switch (MsecTimeout)
    {
    /*
     * No Wait:
     * --------
     * A zero timeout value indicates that we shouldn't wait - just
     * acquire the semaphore if available otherwise return AE_TIME
     * (a.k.a. 'would block').
     */
    case 0:

        if (sem_trywait(Sem) == -1)
        {
            Status = (AE_TIME);
        }
        break;

    /* Wait Indefinitely */

    case ACPI_WAIT_FOREVER:

        if (sem_wait (Sem))
        {
            Status = (AE_TIME);
        }
        break;

    /* Wait with MsecTimeout */

    default:

#ifdef ACPI_USE_ALTERNATE_TIMEOUT
        /*
         * Alternate timeout mechanism for environments where
         * sem_timedwait is not available or does not work properly.
         */
        while (MsecTimeout)
        {
            if (sem_trywait (Sem) == 0)
            {
                /* Got the semaphore */
                return (AE_OK);
            }

            if (MsecTimeout >= 10)
            {
                MsecTimeout -= 10;
                usleep (10 * ACPI_USEC_PER_MSEC); /* ten milliseconds */
            }
            else
            {
                MsecTimeout--;
                usleep (ACPI_USEC_PER_MSEC); /* one millisecond */
            }
```



```
        }
        Status = (AE_TIME);
#else
        /*
         * The interface to sem_timedwait is an absolute time, so we need to
         * get the current time, then add in the millisecond Timeout value.
         */
        if (clock_gettime (CLOCK_REALTIME, &Time) == -1)
        {
            perror ("clock_gettime");
            return (AE_TIME);
        }

        Time.tv_sec += (MsecTimeout / ACPI_MSEC_PER_SEC);
        Time.tv_nsec += ((MsecTimeout % ACPI_MSEC_PER_SEC) * ACPI_NSEC_PER_MSEC)

        /* Handle nanosecond overflow (field must be less than one second) */

        if (Time.tv_nsec >= ACPI_NSEC_PER_SEC)
        {
            Time.tv_sec += (Time.tv_nsec / ACPI_NSEC_PER_SEC);
            Time.tv_nsec = (Time.tv_nsec % ACPI_NSEC_PER_SEC);
        }

        while (((RetVal = sem_timedwait (Sem, &Time)) == -1) && (errno == EINTR)
        {
            continue;
        }

        if (RetVal != 0)
        {
            if (errno != ETIMEDOUT)
            {
                perror ("sem_timedwait");
            }
            Status = (AE_TIME);
        }
#endif
        break;
    }

    return (Status);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsSignalSemaphore
 *
 * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
 *              Units               - Number of units to send
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Send units
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsSignalSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units)
{
    sem_t               *Sem = (sem_t *)Handle;
```

```
916     if (!Sem)
917     {
918         return (AE_BAD_PARAMETER);
919     }

921     if (sem_post (Sem) == -1)
922     {
923         return (AE_LIMIT);
924     }

926     return (AE_OK);
927 }

929 #endif /* ACPI_SINGLE_THREADED */


932 /******************************************************************************
933  *
934  * FUNCTION:    Spinlock interfaces
935  *
936  * DESCRIPTION: Map these interfaces to semaphore interfaces
937  *
938  *****************************************************************************/

940 ACPI_STATUS
941 AcpiOsCreateLock (
942     ACPI_SPINLOCK           *OutHandle)
943 {

945     return (AcpiOsCreateSemaphore (1, 1, OutHandle));
946 }


949 void
950 AcpiOsDeleteLock (
951     ACPI_SPINLOCK           Handle)
952 {
953     AcpiOsDeleteSemaphore (Handle);
954 }


957 ACPI_CPU_FLAGS
958 AcpiOsAcquireLock (
959     ACPI_HANDLE             Handle)
960 {
961     AcpiOsWaitSemaphore (Handle, 1, 0xFFFF);
962     return (0);
963 }


966 void
967 AcpiOsReleaseLock (
968     ACPI_SPINLOCK           Handle,
969     ACPI_CPU_FLAGS          Flags)
970 {
971     AcpiOsSignalSemaphore (Handle, 1);
972 }


975 /******************************************************************************
976  *
977  * FUNCTION:    AcpiOsInstallInterruptHandler
978  *
979  * PARAMETERS:  InterruptNumber     - Level handler should respond to.
980  *              Isr                 - Address of the ACPI interrupt handler
981  *              ExceptPtr           - Where status is returned
```



```
982  *
983  * RETURN:      Handle to the newly installed handler.
984  *
985  * DESCRIPTION: Install an interrupt handler. Used to install the ACPI
986  *              OS-independent handler.
987  *
988  *****************************************************************************/

990 UINT32
991 AcpiOsInstallInterruptHandler (
992     UINT32                  InterruptNumber,
993     ACPI_OSD_HANDLER        ServiceRoutine,
994     void                    *Context)
995 {

997     return (AE_OK);
998 }


1001 /******************************************************************************
1002  *
1003  * FUNCTION:    AcpiOsRemoveInterruptHandler
1004  *
1005  * PARAMETERS:  Handle              - Returned when handler was installed
1006  *
1007  * RETURN:      Status
1008  *
1009  * DESCRIPTION: Uninstalls an interrupt handler.
1010  *
1011  *****************************************************************************/

1013 ACPI_STATUS
1014 AcpiOsRemoveInterruptHandler (
1015     UINT32                  InterruptNumber,
1016     ACPI_OSD_HANDLER        ServiceRoutine)
1017 {

1019     return (AE_OK);
1020 }


1023 /******************************************************************************
1024  *
1025  * FUNCTION:    AcpiOsStall
1026  *
1027  * PARAMETERS:  microseconds        - Time to sleep
1028  *
1029  * RETURN:      Blocks until sleep is completed.
1030  *
1031  * DESCRIPTION: Sleep at microsecond granularity
1032  *
1033  *****************************************************************************/

1035 void
1036 AcpiOsStall (
1037     UINT32                  microseconds)
1038 {

1040     if (microseconds)
1041     {
1042         usleep (microseconds);
1043     }
1044 }


1047 /******************************************************************************
```

```
 *
 * FUNCTION:    AcpiOsSleep
 *
 * PARAMETERS:  milliseconds        - Time to sleep
 *
 * RETURN:      Blocks until sleep is completed.
 *
 * DESCRIPTION: Sleep at millisecond granularity
 *
 *****************************************************************************/

void
AcpiOsSleep (
    UINT64                  milliseconds)
{

    /* Sleep for whole seconds */

    sleep (milliseconds / ACPI_MSEC_PER_SEC);

    /*
     * Sleep for remaining microseconds.
     * Arg to usleep() is in usecs and must be less than 1,000,000 (1 second).
     */
    usleep ((milliseconds % ACPI_MSEC_PER_SEC) * ACPI_USEC_PER_MSEC);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsGetTimer
 *
 * PARAMETERS:  None
 *
 * RETURN:      Current time in 100 nanosecond units
 *
 * DESCRIPTION: Get the current system time
 *
 *****************************************************************************/

UINT64
AcpiOsGetTimer (
    void)
{
    struct timeval          time;


    /* This timer has sufficient resolution for user-space application code */

    gettimeofday (&time, NULL);

    /* (Seconds * 10^7 = 100ns(10^-7)) + (Microseconds(10^-6) * 10^1 = 100ns) */

    return (((UINT64) time.tv_sec * ACPI_100NSEC_PER_SEC) +
            ((UINT64) time.tv_usec * ACPI_100NSEC_PER_USEC));
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsReadPciConfiguration
 *
 * PARAMETERS:  PciId               - Seg/Bus/Dev
 *              Register            - Device Register
 *              Value               - Buffer where value is placed
 *              Width               - Number of bits
```



```
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Read data from PCI configuration space
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsReadPciConfiguration (
    ACPI_PCI_ID             *PciId,
    UINT32                  Register,
    UINT64                  *Value,
    UINT32                  Width)
{

    *Value = 0;
    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsWritePciConfiguration
 *
 * PARAMETERS:  PciId               - Seg/Bus/Dev
 *              Register            - Device Register
 *              Value               - Value to be written
 *              Width               - Number of bits
 *
 * RETURN:      Status.
 *
 * DESCRIPTION: Write data to PCI configuration space
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWritePciConfiguration (
    ACPI_PCI_ID             *PciId,
    UINT32                  Register,
    UINT64                  Value,
    UINT32                  Width)
{

    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsReadPort
 *
 * PARAMETERS:  Address             - Address of I/O port/register to read
 *              Value               - Where value is placed
 *              Width               - Number of bits
 *
 * RETURN:      Value read from port
 *
 * DESCRIPTION: Read data from an I/O port or register
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsReadPort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  *Value,
    UINT32                  Width)
```

```
{

    switch (Width)
    {
    case 8:

        *Value = 0xFF;
        break;

    case 16:

        *Value = 0xFFFF;
        break;

    case 32:

        *Value = 0xFFFFFFFF;
        break;

    default:

        return (AE_BAD_PARAMETER);
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsWritePort
 *
 * PARAMETERS:  Address             - Address of I/O port/register to write
 *              Value               - Value to write
 *              Width               - Number of bits
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write data to an I/O port or register
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWritePort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  Value,
    UINT32                  Width)
{

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsReadMemory
 *
 * PARAMETERS:  Address             - Physical Memory Address to read
 *              Value               - Where value is placed
 *              Width               - Number of bits (8,16,32, or 64)
 *
 * RETURN:      Value read from physical memory address. Always returned
 *              as a 64-bit integer, regardless of the read width.
 *
 * DESCRIPTION: Read data from a physical memory address
 *
```



```
 *****************************************************************************/

ACPI_STATUS
AcpiOsReadMemory (
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT64                  *Value,
    UINT32                  Width)
{

    switch (Width)
    {
    case 8:
    case 16:
    case 32:
    case 64:

        *Value = 0;
        break;

    default:

        return (AE_BAD_PARAMETER);
    }
    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsWriteMemory
 *
 * PARAMETERS:  Address             - Physical Memory Address to write
 *              Value               - Value to write
 *              Width               - Number of bits (8,16,32, or 64)
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write data to a physical memory address
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWriteMemory (
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT64                  Value,
    UINT32                  Width)
{

    return (AE_OK);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsReadable
 *
 * PARAMETERS:  Pointer             - Area to be verified
 *              Length              - Size of area
 *
 * RETURN:      TRUE if readable for entire length
 *
 * DESCRIPTION: Verify that a pointer is valid for reading
 *
 *****************************************************************************/

BOOLEAN
```

```
1312 AcpiOsReadable (
1313     void                    *Pointer,
1314     ACPI_SIZE               Length)
1315 {

1317     return (TRUE);
1318 }


1321 /******************************************************************************
1322  *
1323  * FUNCTION:    AcpiOsWritable
1324  *
1325  * PARAMETERS:  Pointer             - Area to be verified
1326  *              Length              - Size of area
1327  *
1328  * RETURN:      TRUE if writable for entire length
1329  *
1330  * DESCRIPTION: Verify that a pointer is valid for writing
1331  *
1332  *****************************************************************************/

1334 BOOLEAN
1335 AcpiOsWritable (
1336     void                    *Pointer,
1337     ACPI_SIZE               Length)
1338 {

1340     return (TRUE);
1341 }


1344 /******************************************************************************
1345  *
1346  * FUNCTION:    AcpiOsSignal
1347  *
1348  * PARAMETERS:  Function            - ACPI CA signal function code
1349  *              Info                - Pointer to function-dependent structure
1350  *
1351  * RETURN:      Status
1352  *
1353  * DESCRIPTION: Miscellaneous functions. Example implementation only.
1354  *
1355  *****************************************************************************/

1357 ACPI_STATUS
1358 AcpiOsSignal (
1359     UINT32                  Function,
1360     void                    *Info)
1361 {

1363     switch (Function)
1364     {
1365     case ACPI_SIGNAL_FATAL:

1367         break;

1369     case ACPI_SIGNAL_BREAKPOINT:

1371         break;

1373     default:

1375         break;
1376     }
```



```
1378     return (AE_OK);
1379 }

1381 /* Optional multi-thread support */

1383 #ifndef ACPI_SINGLE_THREADED
1384 /******************************************************************************
1385  *
1386  * FUNCTION:    AcpiOsGetThreadId
1387  *
1388  * PARAMETERS:  None
1389  *
1390  * RETURN:      Id of the running thread
1391  *
1392  * DESCRIPTION: Get the ID of the current (running) thread
1393  *
1394  *****************************************************************************/

1396 ACPI_THREAD_ID
1397 AcpiOsGetThreadId (
1398     void)
1399 {
1400     pthread_t               thread;


1403     thread = pthread_self();
1404     return (ACPI_CAST_PTHREAD_T (thread));
1405 }


1408 /******************************************************************************
1409  *
1410  * FUNCTION:    AcpiOsExecute
1411  *
1412  * PARAMETERS:  Type                - Type of execution
1413  *              Function            - Address of the function to execute
1414  *              Context             - Passed as a parameter to the function
1415  *
1416  * RETURN:      Status.
1417  *
1418  * DESCRIPTION: Execute a new thread
1419  *
1420  *****************************************************************************/

1422 ACPI_STATUS
1423 AcpiOsExecute (
1424     ACPI_EXECUTE_TYPE       Type,
1425     ACPI_OSD_EXEC_CALLBACK  Function,
1426     void                    *Context)
1427 {
1428     pthread_t               thread;
1429     int                     ret;


1432     ret = pthread_create (&thread, NULL, (PTHREAD_CALLBACK) Function, Context);
1433     if (ret)
1434     {
1435         AcpiOsPrintf("Create thread failed");
1436     }
1437     return (0);
1438 }

1440 #endif /* ACPI_SINGLE_THREADED */


1443 /******************************************************************************
```

```
1444  *
1445  * FUNCTION:    AcpiOsWaitEventsComplete
1446  *
1447  * PARAMETERS:  None
1448  *
1449  * RETURN:      None
1450  *
1451  * DESCRIPTION: Wait for all asynchronous events to complete. This
1452  *              implementation does nothing.
1453  *
1454  *****************************************************************************/

1456 void
1457 AcpiOsWaitEventsComplete (
1458     void)
1459 {
1460     return;
1461 }
```

```
**********************************************************
    7547 Thu Dec 26 13:50:06 2013
new/usr/src/common/acpica/os_specific/service_layers/oswindir.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: oswindir - Windows directory access interfaces
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include <acpi.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>

typedef struct ExternalFindInfo
{
    struct _finddata_t          DosInfo;
    char                        *FullWildcardSpec;
    long                        FindHandle;
    char                        State;
    char                        RequestedFileType;

} EXTERNAL_FIND_INFO;
```



```
/******************************************************************************
 *
 * FUNCTION:    AcpiOsOpenDirectory
 *
 * PARAMETERS:  DirPathname         - Full pathname to the directory
 *              WildcardSpec        - string of the form "*.c", etc.
 *              RequestedFileType   - Either a directory or normal file
 *
 * RETURN:      A directory "handle" to be used in subsequent search operations.
 *              NULL returned on failure.
 *
 * DESCRIPTION: Open a directory in preparation for a wildcard search
 *
 *****************************************************************************/

void *
AcpiOsOpenDirectory (
    char                    *DirPathname,
    char                    *WildcardSpec,
    char                    RequestedFileType)
{
    long                    FindHandle;
    char                    *FullWildcardSpec;
    EXTERNAL_FIND_INFO      *SearchInfo;


    /* No directory path means "use current directory" - use a dot */

    if (!DirPathname || strlen (DirPathname) == 0)
    {
        DirPathname = ".";
    }

    /* Allocate the info struct that will be returned to the caller */

    SearchInfo = calloc (sizeof (EXTERNAL_FIND_INFO), 1);
    if (!SearchInfo)
    {
        return (NULL);
    }

    /* Allocate space for the full wildcard path */

    FullWildcardSpec = calloc (strlen (DirPathname) + strlen (WildcardSpec) + 2,
    if (!FullWildcardSpec)
    {
        printf ("Could not allocate buffer for wildcard pathname\n");
        return (NULL);
    }

    /* Create the full wildcard path */

    strcpy (FullWildcardSpec, DirPathname);
    strcat (FullWildcardSpec, "/");
    strcat (FullWildcardSpec, WildcardSpec);

    /* Initialize the find functions, get first match */

    FindHandle = _findfirst (FullWildcardSpec, &SearchInfo->DosInfo);
    if (FindHandle == -1)
    {
        /* Failure means that no match was found */

        free (FullWildcardSpec);
        free (SearchInfo);
```

```
127         return (NULL);
128     }

130     /* Save the info in the return structure */

132     SearchInfo->RequestedFileType = RequestedFileType;
133     SearchInfo->FullWildcardSpec = FullWildcardSpec;
134     SearchInfo->FindHandle = FindHandle;
135     SearchInfo->State = 0;
136     return (SearchInfo);
137 }


140 /*****************************************************************************
141  *
142  * FUNCTION:    AcpiOsGetNextFilename
143  *
144  * PARAMETERS:  DirHandle           - Created via AcpiOsOpenDirectory
145  *
146  * RETURN:      Next filename matched. NULL if no more matches.
147  *
148  * DESCRIPTION: Get the next file in the directory that matches the wildcard
149  *              specification.
150  *
151  *****************************************************************************/

153 char *
154 AcpiOsGetNextFilename (
155     void                    *DirHandle)
156 {
157     EXTERNAL_FIND_INFO      *SearchInfo = DirHandle;
158     int                     Status;
159     char                    FileTypeNotMatched = 1;


162     /*
163      * Loop while we have matched files but not found any files of
164      * the requested type.
165      */
166     while (FileTypeNotMatched)
167     {
168         /* On the first call, we already have the first match */

170         if (SearchInfo->State == 0)
171         {
172             /* No longer the first match */

174             SearchInfo->State = 1;
175         }
176         else
177         {
178             /* Get the next match */

180             Status = _findnext (SearchInfo->FindHandle, &SearchInfo->DosInfo);
181             if (Status != 0)
182             {
183                 return (NULL);
184             }
185         }

187         /*
188          * Found a match, now check to make sure that the file type
189          * matches the requested file type (directory or normal file)
190          *
191          * NOTE: use of the attrib field saves us from doing a very
192          * expensive stat() on the file!
```



```
193          */
194         switch (SearchInfo->RequestedFileType)
195         {
196         case REQUEST_FILE_ONLY:

198             /* Anything other than A_SUBDIR is OK */

200             if (!(SearchInfo->DosInfo.attrib & _A_SUBDIR))
201             {
202                 FileTypeNotMatched = 0;
203             }
204             break;

206         case REQUEST_DIR_ONLY:

208             /* Must have A_SUBDIR bit set */

210             if (SearchInfo->DosInfo.attrib & _A_SUBDIR)
211             {
212                 FileTypeNotMatched = 0;
213             }
214             break;

216         default:

218             return (NULL);
219         }
220     }

222     return (SearchInfo->DosInfo.name);
223 }


226 /*****************************************************************************
227  *
228  * FUNCTION:    AcpiOsCloseDirectory
229  *
230  * PARAMETERS:  DirHandle           - Created via AcpiOsOpenDirectory
231  *
232  * RETURN:      None
233  *
234  * DESCRIPTION: Close the open directory and cleanup.
235  *
236  *****************************************************************************/

238 void
239 AcpiOsCloseDirectory (
240     void                    *DirHandle)
241 {
242     EXTERNAL_FIND_INFO      *SearchInfo = DirHandle;


245     /* Close the directory and free allocations */

247     _findclose (SearchInfo->FindHandle);
248     free (SearchInfo->FullWildcardSpec);
249     free (DirHandle);
250 }
```

```
**********************************************************
   12518 Thu Dec 26 13:50:06 2013
new/usr/src/common/acpica/os_specific/service_layers/oswintbl.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: oswintbl - Windows OSL for obtaining ACPI tables
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acutils.h"
#include <stdio.h>

#ifdef WIN32
#pragma warning(disable:4115)   /* warning C4115: (caused by rpcasync.h) */
#include <windows.h>

#elif WIN64
#include <windowsx.h>
#endif

#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("oswintbl")
```



```
/* Local prototypes */

static char *
WindowsFormatException (
    LONG                WinStatus);

/* Globals */

#define LOCAL_BUFFER_SIZE           64

static char             KeyBuffer[LOCAL_BUFFER_SIZE];
static char             ErrorBuffer[LOCAL_BUFFER_SIZE];

/*
 * Tables supported in the Windows registry. SSDTs are not placed into
 * the registry, a limitation.
 */
static char             *SupportedTables[] =
{
    "DSDT",
    "RSDT",
    "FACS",
    "FACP"
};

/* Max index for table above */

#define ACPI_OS_MAX_TABLE_INDEX     3


/******************************************************************************
 *
 * FUNCTION:    WindowsFormatException
 *
 * PARAMETERS:  WinStatus       - Status from a Windows system call
 *
 * RETURN:      Formatted (ascii) exception code. Front-end to Windows
 *              FormatMessage interface.
 *
 * DESCRIPTION: Decode a windows exception
 *
 *****************************************************************************/

static char *
WindowsFormatException (
    LONG                WinStatus)
{

    ErrorBuffer[0] = 0;
    FormatMessage (FORMAT_MESSAGE_FROM_SYSTEM, NULL, WinStatus, 0,
        ErrorBuffer, LOCAL_BUFFER_SIZE, NULL);

    return (ErrorBuffer);
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByAddress
 *
 * PARAMETERS:  Address         - Physical address of the ACPI table
 *              Table           - Where a pointer to the table is returned
 *
 * RETURN:      Status; Table buffer is returned if AE_OK.
 *              AE_NOT_FOUND: A valid table was not found at the address
 *
```

```
 * DESCRIPTION: Get an ACPI table via a physical memory address.
 *
 * NOTE:        Cannot be implemented without a Windows device driver.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByAddress (
    ACPI_PHYSICAL_ADDRESS   Address,
    ACPI_TABLE_HEADER       **Table)
{

    fprintf (stderr, "Get table by address is not supported on Windows\n");
    return (AE_SUPPORT);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByIndex
 *
 * PARAMETERS:  Index           - Which table to get
 *              Table           - Where a pointer to the table is returned
 *              Instance        - Where a pointer to the table instance no. is
 *                                returned
 *              Address         - Where the table physical address is returned
 *
 * RETURN:      Status; Table buffer and physical address returned if AE_OK.
 *              AE_LIMIT: Index is beyond valid limit
 *
 * DESCRIPTION: Get an ACPI table via an index value (0 through n). Returns
 *              AE_LIMIT when an invalid index is reached. Index is not
 *              necessarily an index into the RSDT/XSDT.
 *              Table is obtained from the Windows registry.
 *
 * NOTE:        Cannot get the physical address from the windows registry;
 *              zero is returned instead.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByIndex (
    UINT32                  Index,
    ACPI_TABLE_HEADER       **Table,
    UINT32                  *Instance,
    ACPI_PHYSICAL_ADDRESS   *Address)
{
    ACPI_STATUS             Status;


    if (Index > ACPI_OS_MAX_TABLE_INDEX)
    {
        return (AE_LIMIT);
    }

    Status = AcpiOsGetTableByName (SupportedTables[Index], 0, Table, Address);
    return (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsGetTableByName
 *
 * PARAMETERS:  Signature       - ACPI Signature for desired table. Must be
 *                                a null terminated 4-character string.
```



```
 *              Instance        - For SSDTs (0...n). Use 0 otherwise.
 *              Table           - Where a pointer to the table is returned
 *              Address         - Where the table physical address is returned
 *
 * RETURN:      Status; Table buffer and physical address returned if AE_OK.
 *              AE_LIMIT: Instance is beyond valid limit
 *              AE_NOT_FOUND: A table with the signature was not found
 *
 * DESCRIPTION: Get an ACPI table via a table signature (4 ASCII characters).
 *              Returns AE_LIMIT when an invalid instance is reached.
 *              Table is obtained from the Windows registry.
 *
 * NOTE:        Assumes the input signature is uppercase.
 *              Cannot get the physical address from the windows registry;
 *              zero is returned instead.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsGetTableByName (
    char                    *Signature,
    UINT32                  Instance,
    ACPI_TABLE_HEADER       **Table,
    ACPI_PHYSICAL_ADDRESS   *Address)
{
    HKEY                    Handle = NULL;
    LONG                    WinStatus;
    ULONG                   Type;
    ULONG                   NameSize;
    ULONG                   DataSize;
    HKEY                    SubKey;
    ULONG                   i;
    ACPI_TABLE_HEADER       *ReturnTable;


    /*
     * Windows has no SSDTs in the registry, so multiple instances are
     * not supported.
     */
    if (Instance > 0)
    {
        return (AE_LIMIT);
    }

    /* Get a handle to the table key */

    while (1)
    {
        ACPI_STRCPY (KeyBuffer, "HARDWARE\\ACPI\\");
        if (AcpiUtSafeStrcat (KeyBuffer, sizeof (KeyBuffer), Signature))
        {
            return (AE_BUFFER_OVERFLOW);
        }

        WinStatus = RegOpenKeyEx (HKEY_LOCAL_MACHINE, KeyBuffer,
            0L, KEY_READ, &Handle);

        if (WinStatus != ERROR_SUCCESS)
        {
            /*
             * Somewhere along the way, MS changed the registry entry for
             * the FADT from
             * HARDWARE/ACPI/FACP  to
             * HARDWARE/ACPI/FADT.
             *
             * This code allows for both.
```

```
258              */
259             if (ACPI_COMPARE_NAME (Signature, "FACP"))
260             {
261                 Signature = "FADT";
262             }
263             else if (ACPI_COMPARE_NAME (Signature, "XSDT"))
264             {
265                 Signature = "RSDT";
266             }
267             else
268             {
269                 fprintf (stderr,
270                     "Could not find %s in registry at %s: %s (WinStatus=0x%X)\n"
271                     Signature, KeyBuffer, WindowsFormatException (WinStatus), Wi
272                 return (AE_NOT_FOUND);
273             }
274         }
275         else
276         {
277             break;
278         }
279     }

281     /* Actual data for the table is down a couple levels */

283     for (i = 0; ;)
284     {
285         WinStatus = RegEnumKey (Handle, i, KeyBuffer, sizeof (KeyBuffer));
286         i++;
287         if (WinStatus == ERROR_NO_MORE_ITEMS)
288         {
289             break;
290         }

292         WinStatus = RegOpenKey (Handle, KeyBuffer, &SubKey);
293         if (WinStatus != ERROR_SUCCESS)
294         {
295             fprintf (stderr, "Could not open %s entry: %s\n",
296                 Signature, WindowsFormatException (WinStatus));
297             return (AE_ERROR);
298         }

300         RegCloseKey (Handle);
301         Handle = SubKey;
302         i = 0;
303     }

305     /* Find the (binary) table entry */

307     for (i = 0; ; i++)
308     {
309         NameSize = sizeof (KeyBuffer);
310         WinStatus = RegEnumValue (Handle, i, KeyBuffer, &NameSize, NULL,
311             &Type, NULL, 0);
312         if (WinStatus != ERROR_SUCCESS)
313         {
314             fprintf (stderr, "Could not get %s registry entry: %s\n",
315                 Signature, WindowsFormatException (WinStatus));
316             return (AE_ERROR);
317         }

319         if (Type == REG_BINARY)
320         {
321             break;
322         }
323     }
```



```
325     /* Get the size of the table */

327     WinStatus = RegQueryValueEx (Handle, KeyBuffer, NULL, NULL,
328         NULL, &DataSize);
329     if (WinStatus != ERROR_SUCCESS)
330     {
331         fprintf (stderr, "Could not read the %s table size: %s\n",
332             Signature, WindowsFormatException (WinStatus));
333         return (AE_ERROR);
334     }

336     /* Allocate a new buffer for the table */

338     ReturnTable = malloc (DataSize);
339     if (!ReturnTable)
340     {
341         goto Cleanup;
342     }

344     /* Get the actual table from the registry */

346     WinStatus = RegQueryValueEx (Handle, KeyBuffer, NULL, NULL,
347         (UCHAR *) ReturnTable, &DataSize);
348     if (WinStatus != ERROR_SUCCESS)
349     {
350         fprintf (stderr, "Could not read %s data: %s\n",
351             Signature, WindowsFormatException (WinStatus));
352         free (ReturnTable);
353         return (AE_ERROR);
354     }

356 Cleanup:
357     RegCloseKey (Handle);

359     *Table = ReturnTable;
360     *Address = 0;
361     return (AE_OK);
362 }


365 /* These are here for acpidump only, so we don't need to link oswinxf */

367 #ifdef ACPI_DUMP_APP
368 /******************************************************************************
369  *
370  * FUNCTION:    AcpiOsMapMemory
371  *
372  * PARAMETERS:  Where               - Physical address of memory to be mapped
373  *              Length              - How much memory to map
374  *
375  * RETURN:      Pointer to mapped memory. Null on error.
376  *
377  * DESCRIPTION: Map physical memory into caller's address space
378  *
379  *****************************************************************************/

381 void *
382 AcpiOsMapMemory (
383     ACPI_PHYSICAL_ADDRESS   Where,
384     ACPI_SIZE               Length)
385 {

387     return (ACPI_TO_POINTER ((ACPI_SIZE) Where));
388 }
```

```
391 /******************************************************************************
392  *
393  * FUNCTION:    AcpiOsUnmapMemory
394  *
395  * PARAMETERS:  Where               - Logical address of memory to be unmapped
396  *              Length              - How much memory to unmap
397  *
398  * RETURN:      None.
399  *
400  * DESCRIPTION: Delete a previously created mapping. Where and Length must
401  *              correspond to a previous mapping exactly.
402  *
403  *****************************************************************************/

405 void
406 AcpiOsUnmapMemory (
407     void                    *Where,
408     ACPI_SIZE               Length)
409 {

411     return;
412 }
413 #endif
```

```
**********************************************************
   37033 Thu Dec 26 13:50:06 2013
new/usr/src/common/acpica/os_specific/service_layers/oswinxf.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: oswinxf - Windows OSL
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

#ifdef WIN32
#pragma warning(disable:4115)   /* warning C4115: named type definition in paren

#include <windows.h>
#include <winbase.h>

#elif WIN64
#include <windowsx.h>
#endif

#include <stdio.h>
#include <stdlib.h>
```



```
#include <stdarg.h>
#include <process.h>
#include <time.h>

#define _COMPONENT          ACPI_OS_SERVICES
        ACPI_MODULE_NAME    ("oswinxf")


FILE                        *AcpiGbl_OutputFile;
UINT64                      TimerFrequency;
char                        TableName[ACPI_NAME_SIZE + 1];

#define ACPI_OS_DEBUG_TIMEOUT   30000 /* 30 seconds */


/* Upcalls to AcpiExec application */

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);

void
AeTableOverride (
    ACPI_TABLE_HEADER       *ExistingTable,
    ACPI_TABLE_HEADER       **NewTable);

/*
 * Real semaphores are only used for a multi-threaded application
 */
#ifndef ACPI_SINGLE_THREADED

/* Semaphore information structure */

typedef struct acpi_os_semaphore_info
{
    UINT16                  MaxUnits;
    UINT16                  CurrentUnits;
    void                    *OsHandle;

} ACPI_OS_SEMAPHORE_INFO;

/* Need enough semaphores to run the large aslts suite */

#define ACPI_OS_MAX_SEMAPHORES  256

ACPI_OS_SEMAPHORE_INFO          AcpiGbl_Semaphores[ACPI_OS_MAX_SEMAPHORES];

#endif /* ACPI_SINGLE_THREADED */

BOOLEAN                         AcpiGbl_DebugTimeout = FALSE;

/******************************************************************************
 *
 * FUNCTION:    AcpiOsTerminate
 *
 * PARAMETERS:  None
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Nothing to do for windows
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsTerminate (
    void)
```

```
125 {
126     return (AE_OK);
127 }


130 /******************************************************************************
131  *
132  * FUNCTION:    AcpiOsInitialize
133  *
134  * PARAMETERS:  None
135  *
136  * RETURN:      Status
137  *
138  * DESCRIPTION: Init this OSL
139  *
140  *****************************************************************************/

142 ACPI_STATUS
143 AcpiOsInitialize (
144     void)
145 {
146     LARGE_INTEGER           LocalTimerFrequency;


149 #ifndef ACPI_SINGLE_THREADED
150     /* Clear the semaphore info array */

152     memset (AcpiGbl_Semaphores, 0x00, sizeof (AcpiGbl_Semaphores));
153 #endif

155     AcpiGbl_OutputFile = stdout;

157     /* Get the timer frequency for use in AcpiOsGetTimer */

159     TimerFrequency = 0;
160     if (QueryPerformanceFrequency (&LocalTimerFrequency))
161     {
162         /* Frequency is in ticks per second */

164         TimerFrequency = LocalTimerFrequency.QuadPart;
165     }

167     return (AE_OK);
168 }


171 /******************************************************************************
172  *
173  * FUNCTION:    AcpiOsGetRootPointer
174  *
175  * PARAMETERS:  None
176  *
177  * RETURN:      RSDP physical address
178  *
179  * DESCRIPTION: Gets the root pointer (RSDP)
180  *
181  *****************************************************************************/

183 ACPI_PHYSICAL_ADDRESS
184 AcpiOsGetRootPointer (
185     void)
186 {

188     return (AeLocalGetRootPointer ());
189 }
```



```
192 /******************************************************************************
193  *
194  * FUNCTION:    AcpiOsPredefinedOverride
195  *
196  * PARAMETERS:  InitVal             - Initial value of the predefined object
197  *              NewVal              - The new value for the object
198  *
199  * RETURN:      Status, pointer to value. Null pointer returned if not
200  *              overriding.
201  *
202  * DESCRIPTION: Allow the OS to override predefined names
203  *
204  *****************************************************************************/

206 ACPI_STATUS
207 AcpiOsPredefinedOverride (
208     const ACPI_PREDEFINED_NAMES *InitVal,
209     ACPI_STRING                 *NewVal)
210 {

212     if (!InitVal || !NewVal)
213     {
214         return (AE_BAD_PARAMETER);
215     }

217     *NewVal = NULL;
218     return (AE_OK);
219 }


222 /******************************************************************************
223  *
224  * FUNCTION:    AcpiOsTableOverride
225  *
226  * PARAMETERS:  ExistingTable       - Header of current table (probably firmware
227  *              NewTable            - Where an entire new table is returned.
228  *
229  * RETURN:      Status, pointer to new table. Null pointer returned if no
230  *              table is available to override
231  *
232  * DESCRIPTION: Return a different version of a table if one is available
233  *
234  *****************************************************************************/

236 ACPI_STATUS
237 AcpiOsTableOverride (
238     ACPI_TABLE_HEADER       *ExistingTable,
239     ACPI_TABLE_HEADER       **NewTable)
240 {
241 #ifdef ACPI_ASL_COMPILER
242     ACPI_STATUS             Status;
243     ACPI_PHYSICAL_ADDRESS   Address;
244 #endif

246     if (!ExistingTable || !NewTable)
247     {
248         return (AE_BAD_PARAMETER);
249     }

251     *NewTable = NULL;


254 #ifdef ACPI_EXEC_APP

256     /* Call back up to AcpiExec */
```

```
258     AeTableOverride (ExistingTable, NewTable);
259 #endif


262 #ifdef ACPI_ASL_COMPILER

264     /* Attempt to get the table from the registry */

266     /* Construct a null-terminated string from table signature */

268     ACPI_MOVE_NAME (TableName, ExistingTable->Signature);
269     TableName[ACPI_NAME_SIZE] = 0;

271     Status = AcpiOsGetTableByName (TableName, 0, NewTable, &Address);
272     if (ACPI_SUCCESS (Status))
273     {
274         AcpiOsPrintf ("Table [%s] obtained from registry, %u bytes\n",
275             TableName, (*NewTable)->Length);
276     }
277     else
278     {
279         AcpiOsPrintf ("Could not read table %s from registry (%s)\n",
280             TableName, AcpiFormatException (Status));
281     }
282 #endif

284     return (AE_OK);
285 }


288 /******************************************************************************
289  *
290  * FUNCTION:    AcpiOsPhysicalTableOverride
291  *
292  * PARAMETERS:  ExistingTable       - Header of current table (probably firmware
293  *              NewAddress          - Where new table address is returned
294  *                                    (Physical address)
295  *              NewTableLength      - Where new table length is returned
296  *
297  * RETURN:      Status, address/length of new table. Null pointer returned
298  *              if no table is available to override.
299  *
300  * DESCRIPTION: Returns AE_SUPPORT, function not used in user space.
301  *
302  *****************************************************************************/

304 ACPI_STATUS
305 AcpiOsPhysicalTableOverride (
306     ACPI_TABLE_HEADER       *ExistingTable,
307     ACPI_PHYSICAL_ADDRESS   *NewAddress,
308     UINT32                  *NewTableLength)
309 {

311     return (AE_SUPPORT);
312 }


315 /******************************************************************************
316  *
317  * FUNCTION:    AcpiOsGetTimer
318  *
319  * PARAMETERS:  None
320  *
321  * RETURN:      Current ticks in 100-nanosecond units
322  *
```



```
323  * DESCRIPTION: Get the value of a system timer
324  *
325  *****************************************************************************/

327 UINT64
328 AcpiOsGetTimer (
329     void)
330 {
331     LARGE_INTEGER           Timer;


334     /* Attempt to use hi-granularity timer first */

336     if (TimerFrequency &&
337         QueryPerformanceCounter (&Timer))
338     {
339         /* Convert to 100 nanosecond ticks */

341         return ((UINT64) ((Timer.QuadPart * (UINT64) ACPI_100NSEC_PER_SEC) /
342             TimerFrequency));
343     }

345     /* Fall back to the lo-granularity timer */

347     else
348     {
349         /* Convert milliseconds to 100 nanosecond ticks */

351         return ((UINT64) GetTickCount() * ACPI_100NSEC_PER_MSEC);
352     }
353 }


356 /******************************************************************************
357  *
358  * FUNCTION:    AcpiOsReadable
359  *
360  * PARAMETERS:  Pointer             - Area to be verified
361  *              Length              - Size of area
362  *
363  * RETURN:      TRUE if readable for entire length
364  *
365  * DESCRIPTION: Verify that a pointer is valid for reading
366  *
367  *****************************************************************************/

369 BOOLEAN
370 AcpiOsReadable (
371     void                    *Pointer,
372     ACPI_SIZE               Length)
373 {

375     return ((BOOLEAN) !IsBadReadPtr (Pointer, Length));
376 }


379 /******************************************************************************
380  *
381  * FUNCTION:    AcpiOsWritable
382  *
383  * PARAMETERS:  Pointer             - Area to be verified
384  *              Length              - Size of area
385  *
386  * RETURN:      TRUE if writable for entire length
387  *
388  * DESCRIPTION: Verify that a pointer is valid for writing
```

```
 *
 *****************************************************************************/

BOOLEAN
AcpiOsWritable (
    void                    *Pointer,
    ACPI_SIZE               Length)
{

    return ((BOOLEAN) !IsBadWritePtr (Pointer, Length));
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsRedirectOutput
 *
 * PARAMETERS:  Destination         - An open file handle/pointer
 *
 * RETURN:      None
 *
 * DESCRIPTION: Causes redirect of AcpiOsPrintf and AcpiOsVprintf
 *
 *****************************************************************************/

void
AcpiOsRedirectOutput (
    void                    *Destination)
{

    AcpiGbl_OutputFile = Destination;
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsPrintf
 *
 * PARAMETERS:  Fmt, ...            - Standard printf format
 *
 * RETURN:      None
 *
 * DESCRIPTION: Formatted output
 *
 *****************************************************************************/

void ACPI_INTERNAL_VAR_XFACE
AcpiOsPrintf (
    const char              *Fmt,
    ...)
{
    va_list                 Args;
    UINT8                   Flags;


    Flags = AcpiGbl_DbOutputFlags;
    if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
    {
        /* Output is directable to either a file (if open) or the console */

        if (AcpiGbl_DebugFile)
        {
            /* Output file is open, send the output there */

            va_start (Args, Fmt);
            vfprintf (AcpiGbl_DebugFile, Fmt, Args);
```



```
            va_end (Args);
        }
        else
        {
            /* No redirection, send output to console (once only!) */

            Flags |= ACPI_DB_CONSOLE_OUTPUT;
        }
    }

    if (Flags & ACPI_DB_CONSOLE_OUTPUT)
    {
        va_start (Args, Fmt);
        vfprintf (AcpiGbl_OutputFile, Fmt, Args);
        va_end (Args);
    }

    return;
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsVprintf
 *
 * PARAMETERS:  Fmt                 - Standard printf format
 *              Args                - Argument list
 *
 * RETURN:      None
 *
 * DESCRIPTION: Formatted output with argument list pointer
 *
 *****************************************************************************/

void
AcpiOsVprintf (
    const char              *Fmt,
    va_list                 Args)
{
    INT32                   Count = 0;
    UINT8                   Flags;


    Flags = AcpiGbl_DbOutputFlags;
    if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
    {
        /* Output is directable to either a file (if open) or the console */

        if (AcpiGbl_DebugFile)
        {
            /* Output file is open, send the output there */

            Count = vfprintf (AcpiGbl_DebugFile, Fmt, Args);
        }
        else
        {
            /* No redirection, send output to console (once only!) */

            Flags |= ACPI_DB_CONSOLE_OUTPUT;
        }
    }

    if (Flags & ACPI_DB_CONSOLE_OUTPUT)
    {
        Count = vfprintf (AcpiGbl_OutputFile, Fmt, Args);
    }
```

```
522     return;
523 }


526 /******************************************************************************
527  *
528  * FUNCTION:    AcpiOsGetLine
529  *
530  * PARAMETERS:  Buffer              - Where to return the command line
531  *              BufferLength        - Maximum length of Buffer
532  *              BytesRead           - Where the actual byte count is returned
533  *
534  * RETURN:      Status and actual bytes read
535  *
536  * DESCRIPTION: Formatted input with argument list pointer
537  *
538  *****************************************************************************/

540 ACPI_STATUS
541 AcpiOsGetLine (
542     char                    *Buffer,
543     UINT32                  BufferLength,
544     UINT32                  *BytesRead)
545 {
546     int                     Temp;
547     UINT32                  i;


550     for (i = 0; ; i++)
551     {
552         if (i >= BufferLength)
553         {
554             return (AE_BUFFER_OVERFLOW);
555         }

557         if ((Temp = getchar ()) == EOF)
558         {
559             return (AE_ERROR);
560         }

562         if (!Temp || Temp == '\n')
563         {
564             break;
565         }

567         Buffer [i] = (char) Temp;
568     }

570     /* Null terminate the buffer */

572     Buffer [i] = 0;

574     /* Return the number of bytes in the string */

576     if (BytesRead)
577     {
578         *BytesRead = i;
579     }
580     return (AE_OK);
581 }


584 /******************************************************************************
585  *
586  * FUNCTION:    AcpiOsMapMemory
```



```
587  *
588  * PARAMETERS:  Where               - Physical address of memory to be mapped
589  *              Length              - How much memory to map
590  *
591  * RETURN:      Pointer to mapped memory. Null on error.
592  *
593  * DESCRIPTION: Map physical memory into caller's address space
594  *
595  *****************************************************************************/

597 void *
598 AcpiOsMapMemory (
599     ACPI_PHYSICAL_ADDRESS   Where,
600     ACPI_SIZE               Length)
601 {

603     return (ACPI_TO_POINTER ((ACPI_SIZE) Where));
604 }


607 /******************************************************************************
608  *
609  * FUNCTION:    AcpiOsUnmapMemory
610  *
611  * PARAMETERS:  Where               - Logical address of memory to be unmapped
612  *              Length              - How much memory to unmap
613  *
614  * RETURN:      None.
615  *
616  * DESCRIPTION: Delete a previously created mapping. Where and Length must
617  *              correspond to a previous mapping exactly.
618  *
619  *****************************************************************************/

621 void
622 AcpiOsUnmapMemory (
623     void                    *Where,
624     ACPI_SIZE               Length)
625 {

627     return;
628 }


631 /******************************************************************************
632  *
633  * FUNCTION:    AcpiOsAllocate
634  *
635  * PARAMETERS:  Size                - Amount to allocate, in bytes
636  *
637  * RETURN:      Pointer to the new allocation. Null on error.
638  *
639  * DESCRIPTION: Allocate memory. Algorithm is dependent on the OS.
640  *
641  *****************************************************************************/

643 void *
644 AcpiOsAllocate (
645     ACPI_SIZE               Size)
646 {
647     void                    *Mem;


650     Mem = (void *) malloc ((size_t) Size);

652     return (Mem);
```

```
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsFree
 *
 * PARAMETERS:  Mem                 - Pointer to previously allocated memory
 *
 * RETURN:      None.
 *
 * DESCRIPTION: Free memory allocated via AcpiOsAllocate
 *
 *****************************************************************************/

void
AcpiOsFree (
    void                    *Mem)
{

    free (Mem);
}


#ifdef ACPI_SINGLE_THREADED
/******************************************************************************
 *
 * FUNCTION:    Semaphore stub functions
 *
 * DESCRIPTION: Stub functions used for single-thread applications that do
 *              not require semaphore synchronization. Full implementations
 *              of these functions appear after the stubs.
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsCreateSemaphore (
    UINT32              MaxUnits,
    UINT32              InitialUnits,
    ACPI_HANDLE         *OutHandle)
{
    *OutHandle = (ACPI_HANDLE) 1;
    return (AE_OK);
}

ACPI_STATUS
AcpiOsDeleteSemaphore (
    ACPI_HANDLE         Handle)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiOsWaitSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units,
    UINT16              Timeout)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiOsSignalSemaphore (
    ACPI_HANDLE         Handle,
    UINT32              Units)
{
```



```
    return (AE_OK);
}

#else
/******************************************************************************
 *
 * FUNCTION:    AcpiOsCreateSemaphore
 *
 * PARAMETERS:  MaxUnits            - Maximum units that can be sent
 *              InitialUnits        - Units to be assigned to the new semaphore
 *              OutHandle           - Where a handle will be returned
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Create an OS semaphore
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsCreateSemaphore (
    UINT32              MaxUnits,
    UINT32              InitialUnits,
    ACPI_SEMAPHORE      *OutHandle)
{
    void                *Mutex;
    UINT32              i;

    ACPI_FUNCTION_NAME (OsCreateSemaphore);


    if (MaxUnits == ACPI_UINT32_MAX)
    {
        MaxUnits = 255;
    }

    if (InitialUnits == ACPI_UINT32_MAX)
    {
        InitialUnits = MaxUnits;
    }

    if (InitialUnits > MaxUnits)
    {
        return (AE_BAD_PARAMETER);
    }

    /* Find an empty slot */

    for (i = 0; i < ACPI_OS_MAX_SEMAPHORES; i++)
    {
        if (!AcpiGbl_Semaphores[i].OsHandle)
        {
            break;
        }
    }
    if (i >= ACPI_OS_MAX_SEMAPHORES)
    {
        ACPI_EXCEPTION ((AE_INFO, AE_LIMIT,
            "Reached max semaphores (%u), could not create", ACPI_OS_MAX_SEMAPHO
        return (AE_LIMIT);
    }

    /* Create an OS semaphore */

    Mutex = CreateSemaphore (NULL, InitialUnits, MaxUnits, NULL);
    if (!Mutex)
    {
```

```
785         ACPI_ERROR ((AE_INFO, "Could not create semaphore"));
786         return (AE_NO_MEMORY);
787     }

789     AcpiGbl_Semaphores[i].MaxUnits = (UINT16) MaxUnits;
790     AcpiGbl_Semaphores[i].CurrentUnits = (UINT16) InitialUnits;
791     AcpiGbl_Semaphores[i].OsHandle = Mutex;

793     ACPI_DEBUG_PRINT ((ACPI_DB_MUTEX, "Handle=%u, Max=%u, Current=%u, OsHandle=%
794             i, MaxUnits, InitialUnits, Mutex));

796     *OutHandle = (void *) i;
797     return (AE_OK);
798 }


801 /******************************************************************************
802  *
803  * FUNCTION:    AcpiOsDeleteSemaphore
804  *
805  * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
806  *
807  * RETURN:      Status
808  *
809  * DESCRIPTION: Delete an OS semaphore
810  *
811  *****************************************************************************/

813 ACPI_STATUS
814 AcpiOsDeleteSemaphore (
815     ACPI_SEMAPHORE      Handle)
816 {
817     UINT32              Index = (UINT32) Handle;


820     if ((Index >= ACPI_OS_MAX_SEMAPHORES) ||
821         !AcpiGbl_Semaphores[Index].OsHandle)
822     {
823         return (AE_BAD_PARAMETER);
824     }

826     CloseHandle (AcpiGbl_Semaphores[Index].OsHandle);
827     AcpiGbl_Semaphores[Index].OsHandle = NULL;
828     return (AE_OK);
829 }


832 /******************************************************************************
833  *
834  * FUNCTION:    AcpiOsWaitSemaphore
835  *
836  * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
837  *              Units               - How many units to wait for
838  *              Timeout             - How long to wait
839  *
840  * RETURN:      Status
841  *
842  * DESCRIPTION: Wait for units
843  *
844  *****************************************************************************/

846 ACPI_STATUS
847 AcpiOsWaitSemaphore (
848     ACPI_SEMAPHORE      Handle,
849     UINT32              Units,
850     UINT16              Timeout)
```



```
851 {
852     UINT32              Index = (UINT32) Handle;
853     UINT32              WaitStatus;
854     UINT32              OsTimeout = Timeout;


857     ACPI_FUNCTION_ENTRY ();


860     if ((Index >= ACPI_OS_MAX_SEMAPHORES) ||
861         !AcpiGbl_Semaphores[Index].OsHandle)
862     {
863         return (AE_BAD_PARAMETER);
864     }

866     if (Units > 1)
867     {
868         printf ("WaitSemaphore: Attempt to receive %u units\n", Units);
869         return (AE_NOT_IMPLEMENTED);
870     }

872     if (Timeout == ACPI_WAIT_FOREVER)
873     {
874         OsTimeout = INFINITE;
875         if (AcpiGbl_DebugTimeout)
876         {
877             /* The debug timeout will prevent hang conditions */

879             OsTimeout = ACPI_OS_DEBUG_TIMEOUT;
880         }
881     }
882     else
883     {
884         /* Add 10ms to account for clock tick granularity */

886         OsTimeout += 10;
887     }

889     WaitStatus = WaitForSingleObject (AcpiGbl_Semaphores[Index].OsHandle, OsTime
890     if (WaitStatus == WAIT_TIMEOUT)
891     {
892         if (AcpiGbl_DebugTimeout)
893         {
894             ACPI_EXCEPTION ((AE_INFO, AE_TIME,
895                 "Debug timeout on semaphore 0x%04X (%ums)\n",
896                 Index, ACPI_OS_DEBUG_TIMEOUT));
897         }
898         return (AE_TIME);
899     }

901     if (AcpiGbl_Semaphores[Index].CurrentUnits == 0)
902     {
903         ACPI_ERROR ((AE_INFO, "%s - No unit received. Timeout 0x%X, OS_Status 0x
904             AcpiUtGetMutexName (Index), Timeout, WaitStatus));

906         return (AE_OK);
907     }

909     AcpiGbl_Semaphores[Index].CurrentUnits--;
910     return (AE_OK);
911 }


914 /******************************************************************************
915  *
916  * FUNCTION:    AcpiOsSignalSemaphore
```

```
 *
 * PARAMETERS:  Handle              - Handle returned by AcpiOsCreateSemaphore
 *              Units               - Number of units to send
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Send units
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsSignalSemaphore (
    ACPI_SEMAPHORE      Handle,
    UINT32              Units)
{
    UINT32              Index = (UINT32) Handle;


    ACPI_FUNCTION_ENTRY ();


    if (Index >= ACPI_OS_MAX_SEMAPHORES)
    {
        printf ("SignalSemaphore: Index/Handle out of range: %2.2X\n", Index);
        return (AE_BAD_PARAMETER);
    }

    if (!AcpiGbl_Semaphores[Index].OsHandle)
    {
        printf ("SignalSemaphore: Null OS handle, Index %2.2X\n", Index);
        return (AE_BAD_PARAMETER);
    }

    if (Units > 1)
    {
        printf ("SignalSemaphore: Attempt to signal %u units, Index %2.2X\n", Un
        return (AE_NOT_IMPLEMENTED);
    }

    if ((AcpiGbl_Semaphores[Index].CurrentUnits + 1) >
        AcpiGbl_Semaphores[Index].MaxUnits)
    {
        ACPI_ERROR ((AE_INFO,
            "Oversignalled semaphore[%u]! Current %u Max %u",
            Index, AcpiGbl_Semaphores[Index].CurrentUnits,
            AcpiGbl_Semaphores[Index].MaxUnits));

        return (AE_LIMIT);
    }

    AcpiGbl_Semaphores[Index].CurrentUnits++;
    ReleaseSemaphore (AcpiGbl_Semaphores[Index].OsHandle, Units, NULL);

    return (AE_OK);
}

#endif /* ACPI_SINGLE_THREADED */


/*****************************************************************************
 *
 * FUNCTION:    Spinlock interfaces
 *
 * DESCRIPTION: Map these interfaces to semaphore interfaces
 *
 *****************************************************************************/
```



```
ACPI_STATUS
AcpiOsCreateLock (
    ACPI_SPINLOCK           *OutHandle)
{
    return (AcpiOsCreateSemaphore (1, 1, OutHandle));
}

void
AcpiOsDeleteLock (
    ACPI_SPINLOCK           Handle)
{
    AcpiOsDeleteSemaphore (Handle);
}

ACPI_CPU_FLAGS
AcpiOsAcquireLock (
    ACPI_SPINLOCK           Handle)
{
    AcpiOsWaitSemaphore (Handle, 1, 0xFFFF);
    return (0);
}

void
AcpiOsReleaseLock (
    ACPI_SPINLOCK           Handle,
    ACPI_CPU_FLAGS          Flags)
{
    AcpiOsSignalSemaphore (Handle, 1);
}


#if ACPI_FUTURE_IMPLEMENTATION

/* Mutex interfaces, just implement with a semaphore */

ACPI_STATUS
AcpiOsCreateMutex (
    ACPI_MUTEX              *OutHandle)
{
    return (AcpiOsCreateSemaphore (1, 1, OutHandle));
}

void
AcpiOsDeleteMutex (
    ACPI_MUTEX              Handle)
{
    AcpiOsDeleteSemaphore (Handle);
}

ACPI_STATUS
AcpiOsAcquireMutex (
    ACPI_MUTEX              Handle,
    UINT16                  Timeout)
{
    AcpiOsWaitSemaphore (Handle, 1, Timeout);
    return (0);
}

void
AcpiOsReleaseMutex (
    ACPI_MUTEX              Handle)
{
    AcpiOsSignalSemaphore (Handle, 1);
}
#endif
```

```
1051 /******************************************************************************
1052  *
1053  * FUNCTION:    AcpiOsInstallInterruptHandler
1054  *
1055  * PARAMETERS:  InterruptNumber     - Level handler should respond to.
1056  *              ServiceRoutine      - Address of the ACPI interrupt handler
1057  *              Context             - User context
1058  *
1059  * RETURN:      Handle to the newly installed handler.
1060  *
1061  * DESCRIPTION: Install an interrupt handler. Used to install the ACPI
1062  *              OS-independent handler.
1063  *
1064  *****************************************************************************/

1066 UINT32
1067 AcpiOsInstallInterruptHandler (
1068     UINT32                  InterruptNumber,
1069     ACPI_OSD_HANDLER        ServiceRoutine,
1070     void                    *Context)
1071 {

1073     return (AE_OK);
1074 }


1077 /******************************************************************************
1078  *
1079  * FUNCTION:    AcpiOsRemoveInterruptHandler
1080  *
1081  * PARAMETERS:  Handle              - Returned when handler was installed
1082  *
1083  * RETURN:      Status
1084  *
1085  * DESCRIPTION: Uninstalls an interrupt handler.
1086  *
1087  *****************************************************************************/

1089 ACPI_STATUS
1090 AcpiOsRemoveInterruptHandler (
1091     UINT32                  InterruptNumber,
1092     ACPI_OSD_HANDLER        ServiceRoutine)
1093 {

1095     return (AE_OK);
1096 }


1099 /******************************************************************************
1100  *
1101  * FUNCTION:    AcpiOsStall
1102  *
1103  * PARAMETERS:  Microseconds        - Time to stall
1104  *
1105  * RETURN:      None. Blocks until stall is completed.
1106  *
1107  * DESCRIPTION: Sleep at microsecond granularity
1108  *
1109  *****************************************************************************/

1111 void
1112 AcpiOsStall (
1113     UINT32                  Microseconds)
1114 {
```



```
1116     Sleep ((Microseconds / ACPI_USEC_PER_MSEC) + 1);
1117     return;
1118 }


1121 /******************************************************************************
1122  *
1123  * FUNCTION:    AcpiOsSleep
1124  *
1125  * PARAMETERS:  Milliseconds        - Time to sleep
1126  *
1127  * RETURN:      None. Blocks until sleep is completed.
1128  *
1129  * DESCRIPTION: Sleep at millisecond granularity
1130  *
1131  *****************************************************************************/

1133 void
1134 AcpiOsSleep (
1135     UINT64                  Milliseconds)
1136 {

1138     /* Add 10ms to account for clock tick granularity */

1140     Sleep (((unsigned long) Milliseconds) + 10);
1141     return;
1142 }


1145 /******************************************************************************
1146  *
1147  * FUNCTION:    AcpiOsReadPciConfiguration
1148  *
1149  * PARAMETERS:  PciId               - Seg/Bus/Dev
1150  *              Register            - Device Register
1151  *              Value               - Buffer where value is placed
1152  *              Width               - Number of bits
1153  *
1154  * RETURN:      Status
1155  *
1156  * DESCRIPTION: Read data from PCI configuration space
1157  *
1158  *****************************************************************************/

1160 ACPI_STATUS
1161 AcpiOsReadPciConfiguration (
1162     ACPI_PCI_ID             *PciId,
1163     UINT32                  Register,
1164     UINT64                  *Value,
1165     UINT32                  Width)
1166 {

1168     *Value = 0;
1169     return (AE_OK);
1170 }


1173 /******************************************************************************
1174  *
1175  * FUNCTION:    AcpiOsWritePciConfiguration
1176  *
1177  * PARAMETERS:  PciId               - Seg/Bus/Dev
1178  *              Register            - Device Register
1179  *              Value               - Value to be written
1180  *              Width               - Number of bits
```

```
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Write data to PCI configuration space
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWritePciConfiguration (
    ACPI_PCI_ID             *PciId,
    UINT32                  Register,
    UINT64                  Value,
    UINT32                  Width)
{

    return (AE_OK);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsReadPort
 *
 * PARAMETERS:  Address             - Address of I/O port/register to read
 *              Value               - Where value is placed
 *              Width               - Number of bits
 *
 * RETURN:      Value read from port
 *
 * DESCRIPTION: Read data from an I/O port or register
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsReadPort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  *Value,
    UINT32                  Width)
{
    ACPI_FUNCTION_NAME (OsReadPort);


    switch (Width)
    {
    case 8:

        *Value = 0xFF;
        break;

    case 16:

        *Value = 0xFFFF;
        break;

    case 32:

        *Value = 0xFFFFFFFF;
        break;

    default:

        ACPI_ERROR ((AE_INFO, "Bad width parameter: %X", Width));
        return (AE_BAD_PARAMETER);
    }

    return (AE_OK);
```



```
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsWritePort
 *
 * PARAMETERS:  Address             - Address of I/O port/register to write
 *              Value               - Value to write
 *              Width               - Number of bits
 *
 * RETURN:      None
 *
 * DESCRIPTION: Write data to an I/O port or register
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsWritePort (
    ACPI_IO_ADDRESS         Address,
    UINT32                  Value,
    UINT32                  Width)
{
    ACPI_FUNCTION_NAME (OsWritePort);


    if ((Width == 8) || (Width == 16) || (Width == 32))
    {
        return (AE_OK);
    }

    ACPI_ERROR ((AE_INFO, "Bad width parameter: %X", Width));
    return (AE_BAD_PARAMETER);
}


/*****************************************************************************
 *
 * FUNCTION:    AcpiOsReadMemory
 *
 * PARAMETERS:  Address             - Physical Memory Address to read
 *              Value               - Where value is placed
 *              Width               - Number of bits (8,16,32, or 64)
 *
 * RETURN:      Value read from physical memory address. Always returned
 *              as a 64-bit integer, regardless of the read width.
 *
 * DESCRIPTION: Read data from a physical memory address
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsReadMemory (
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT64                  *Value,
    UINT32                  Width)
{

    switch (Width)
    {
    case 8:
    case 16:
    case 32:
    case 64:

        *Value = 0;
```

```
1313         break;

1315     default:

1317         return (AE_BAD_PARAMETER);
1318         break;
1319     }

1321     return (AE_OK);
1322 }


1325 /******************************************************************************
1326  *
1327  * FUNCTION:    AcpiOsWriteMemory
1328  *
1329  * PARAMETERS:  Address             - Physical Memory Address to write
1330  *              Value               - Value to write
1331  *              Width               - Number of bits (8,16,32, or 64)
1332  *
1333  * RETURN:      None
1334  *
1335  * DESCRIPTION: Write data to a physical memory address
1336  *
1337  *****************************************************************************/

1339 ACPI_STATUS
1340 AcpiOsWriteMemory (
1341     ACPI_PHYSICAL_ADDRESS   Address,
1342     UINT64                  Value,
1343     UINT32                  Width)
1344 {

1346     return (AE_OK);
1347 }


1350 /******************************************************************************
1351  *
1352  * FUNCTION:    AcpiOsSignal
1353  *
1354  * PARAMETERS:  Function            - ACPI CA signal function code
1355  *              Info                - Pointer to function-dependent structure
1356  *
1357  * RETURN:      Status
1358  *
1359  * DESCRIPTION: Miscellaneous functions. Example implementation only.
1360  *
1361  *****************************************************************************/

1363 ACPI_STATUS
1364 AcpiOsSignal (
1365     UINT32                  Function,
1366     void                    *Info)
1367 {

1369     switch (Function)
1370     {
1371     case ACPI_SIGNAL_FATAL:

1373         break;

1375     case ACPI_SIGNAL_BREAKPOINT:

1377         break;
```



```
1379     default:

1381         break;
1382     }

1384     return (AE_OK);
1385 }


1388 /******************************************************************************
1389  *
1390  * FUNCTION:    Local cache interfaces
1391  *
1392  * DESCRIPTION: Implements cache interfaces via malloc/free for testing
1393  *              purposes only.
1394  *
1395  *****************************************************************************/

1397 #ifndef ACPI_USE_LOCAL_CACHE

1399 ACPI_STATUS
1400 AcpiOsCreateCache (
1401     char                    *CacheName,
1402     UINT16                  ObjectSize,
1403     UINT16                  MaxDepth,
1404     ACPI_CACHE_T            **ReturnCache)
1405 {
1406     ACPI_MEMORY_LIST        *NewCache;


1409     NewCache = malloc (sizeof (ACPI_MEMORY_LIST));
1410     if (!NewCache)
1411     {
1412         return (AE_NO_MEMORY);
1413     }

1415     memset (NewCache, 0, sizeof (ACPI_MEMORY_LIST));
1416     NewCache->ListName = CacheName;
1417     NewCache->ObjectSize = ObjectSize;
1418     NewCache->MaxDepth = MaxDepth;

1420     *ReturnCache = (ACPI_CACHE_T) NewCache;
1421     return (AE_OK);
1422 }

1424 ACPI_STATUS
1425 AcpiOsDeleteCache (
1426     ACPI_CACHE_T            *Cache)
1427 {
1428     free (Cache);
1429     return (AE_OK);
1430 }

1432 ACPI_STATUS
1433 AcpiOsPurgeCache (
1434     ACPI_CACHE_T            *Cache)
1435 {
1436     return (AE_OK);
1437 }

1439 void *
1440 AcpiOsAcquireObject (
1441     ACPI_CACHE_T            *Cache)
1442 {
1443     void                    *NewObject;
```

```
    NewObject = malloc (((ACPI_MEMORY_LIST *) Cache)->ObjectSize);
    memset (NewObject, 0, ((ACPI_MEMORY_LIST *) Cache)->ObjectSize);

    return (NewObject);
}

ACPI_STATUS
AcpiOsReleaseObject (
    ACPI_CACHE_T            *Cache,
    void                    *Object)
{
    free (Object);
    return (AE_OK);
}

#endif /* ACPI_USE_LOCAL_CACHE */


/* Optional multi-thread support */

#ifndef ACPI_SINGLE_THREADED
/******************************************************************************
 *
 * FUNCTION:    AcpiOsGetThreadId
 *
 * PARAMETERS:  None
 *
 * RETURN:      Id of the running thread
 *
 * DESCRIPTION: Get the Id of the current (running) thread
 *
 *****************************************************************************/

ACPI_THREAD_ID
AcpiOsGetThreadId (
    void)
{
    DWORD                   ThreadId;

    /* Ensure ID is never 0 */

    ThreadId = GetCurrentThreadId ();
    return ((ACPI_THREAD_ID) (ThreadId + 1));
}


/******************************************************************************
 *
 * FUNCTION:    AcpiOsExecute
 *
 * PARAMETERS:  Type                - Type of execution
 *              Function            - Address of the function to execute
 *              Context             - Passed as a parameter to the function
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Execute a new thread
 *
 *****************************************************************************/

ACPI_STATUS
AcpiOsExecute (
    ACPI_EXECUTE_TYPE       Type,
    ACPI_OSD_EXEC_CALLBACK  Function,
    void                    *Context)
{
```



```
    _beginthread (Function, (unsigned) 0, Context);
    return (0);
}

#endif /* ACPI_SINGLE_THREADED */


/******************************************************************************
 *
 * FUNCTION:    AcpiOsWaitEventsComplete
 *
 * PARAMETERS:  None
 *
 * RETURN:      None
 *
 * DESCRIPTION: Wait for all asynchronous events to complete. This
 *              implementation does nothing.
 *
 *****************************************************************************/

void
AcpiOsWaitEventsComplete (
    void)
{
    return;
}
```

```
**********************************************************
   21798 Thu Dec 26 13:50:07 2013
new/usr/src/common/acpica/tools/acpibin/abcompare.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: abcompare - compare AML files
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpibin.h"


FILE                        *File1;
FILE                        *File2;
ACPI_TABLE_HEADER           Header1;
ACPI_TABLE_HEADER           Header2;

#define BUFFER_SIZE         256
char                        Buffer[BUFFER_SIZE];


/* Local prototypes */

static BOOLEAN
AbValidateHeader (
    ACPI_TABLE_HEADER       *Header);
```



```
static UINT8
AcpiTbSumTable (
    void                    *Buffer,
    UINT32                  Length);

static char *
AbGetFile (
    char                    *Filename,
    UINT32                  *FileSize);

static UINT32
AbGetFileSize (
    FILE                    *File);

static void
AbPrintHeaderInfo (
    ACPI_TABLE_HEADER       *Header);

static void
AbPrintHeadersInfo (
    ACPI_TABLE_HEADER       *Header,
    ACPI_TABLE_HEADER       *Header2);

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);


/******************************************************************************
 *
 * FUNCTION:    UtHexCharToValue
 *
 * PARAMETERS:  HexChar         - Hex character in Ascii
 *
 * RETURN:      The binary value of the hex character
 *
 * DESCRIPTION: Perform ascii-to-hex translation
 *
 *****************************************************************************/

static UINT8
UtHexCharToValue (
    int                     HexChar,
    UINT8                   *OutBinary)
{

    if (HexChar >= 0x30 && HexChar <= 0x39)
    {
        *OutBinary = (UINT8) (HexChar - 0x30);
        return (1);
    }

    else if (HexChar >= 0x41 && HexChar <= 0x46)
    {
        *OutBinary = (UINT8) (HexChar - 0x37);
        return (1);
    }

    else if (HexChar >= 0x61 && HexChar <= 0x66)
    {
        *OutBinary = (UINT8) (HexChar - 0x57);
        return (1);
    }
    return (0);
}
```

```
128 static UINT8
129 AbHexByteToBinary (
130     char                    *HexString,
131     char                    *OutBinary)
132 {
133     UINT8                   Local1;
134     UINT8                   Local2;


137     if (!UtHexCharToValue (HexString[0], &Local1))
138     {
139         return (0);
140     }
141     if (!UtHexCharToValue (HexString[1], &Local2))
142     {
143         return (0);
144     }

146     *OutBinary = (UINT8) ((Local1 << 4) | Local2);
147     return (2);

149 }


152 /******************************************************************************
153  *
154  * FUNCTION:    AbValidateHeader
155  *
156  * DESCRIPTION: Check for valid ACPI table header
157  *
158  *****************************************************************************/

160 static BOOLEAN
161 AbValidateHeader (
162     ACPI_TABLE_HEADER       *Header)
163 {

165     if (!AcpiUtValidAcpiName (Header->Signature))
166     {
167         printf ("Header signature is invalid\n");
168         return (FALSE);
169     }

171     return (TRUE);
172 }


175 /******************************************************************************
176  *
177  * FUNCTION:    AcpiTbSumTable
178  *
179  * PARAMETERS:  Buffer              - Buffer to checksum
180  *              Length              - Size of the buffer
181  *
182  * RETURNS      8 bit checksum of buffer
183  *
184  * DESCRIPTION: Computes an 8 bit checksum of the buffer(length) and returns it.
185  *
186  *****************************************************************************/

188 static UINT8
189 AcpiTbSumTable (
190     void                    *Buffer,
191     UINT32                  Length)
192 {
193     const UINT8             *limit;
```



```
194     const UINT8             *rover;
195     UINT8                   sum = 0;


198     if (Buffer && Length)
199     {
200         /* Buffer and Length are valid */

202         limit = (UINT8 *) Buffer + Length;

204         for (rover = Buffer; rover < limit; rover++)
205         {
206             sum = (UINT8) (sum + *rover);
207         }
208     }
209     return (sum);
210 }


213 /******************************************************************************
214  *
215  * FUNCTION:    AbPrintHeaderInfo
216  *
217  * PARAMETERS:  Header              - An ACPI table header
218  *
219  * RETURNS      None.
220  *
221  * DESCRIPTION: Format and display header contents.
222  *
223  *****************************************************************************/

225 static void
226 AbPrintHeaderInfo (
227     ACPI_TABLE_HEADER       *Header)
228 {

230     /* Display header information */

232     printf ("Signature         : %4.4s\n",    Header->Signature);
233     printf ("Length            : %8.8X\n",    Header->Length);
234     printf ("Revision          : %2.2X\n",    Header->Revision);
235     printf ("Checksum          : %2.2X\n",    Header->Checksum);
236     printf ("OEM ID            : %6.6s\n",    Header->OemId);
237     printf ("OEM Table ID      : %8.8s\n",    Header->OemTableId);
238     printf ("OEM Revision      : %8.8X\n",    Header->OemRevision);
239     printf ("ASL Compiler ID   : %4.4s\n",    Header->AslCompilerId);
240     printf ("Compiler Revision : %8.8X\n",    Header->AslCompilerRevision);
241     printf ("\n");
242 }

244 static void
245 AbPrintHeadersInfo (
246     ACPI_TABLE_HEADER       *Header,
247     ACPI_TABLE_HEADER       *Header2)
248 {

250     /* Display header information for both headers */

252     printf ("Signature         %8.4s : %4.4s\n",    Header->Signature, Header2-
253     printf ("Length            %8.8X : %8.8X\n",    Header->Length, Header2->Le
254     printf ("Revision          %8.2X : %2.2X\n",    Header->Revision, Header2->
255     printf ("Checksum          %8.2X : %2.2X\n",    Header->Checksum, Header2->
256     printf ("OEM ID            %8.6s : %6.6s\n",    Header->OemId, Header2->Oem
257     printf ("OEM Table ID      %8.8s : %8.8s\n",    Header->OemTableId, Header2
258     printf ("OEM Revision      %8.8X : %8.8X\n",    Header->OemRevision, Header
259     printf ("ASL Compiler ID   %8.4s : %4.4s\n",    Header->AslCompilerId, Head
```

```
260     printf ("Compiler Revision  %8.8X : %8.8X\n",    Header->AslCompilerRevision
261     printf ("\n");
262 }


265 /******************************************************************************
266  *
267  * FUNCTION:    AbDisplayHeader
268  *
269  * DESCRIPTION: Display an ACPI table header
270  *
271  *****************************************************************************/

273 void
274 AbDisplayHeader (
275     char                    *File1Path)
276 {
277     UINT32                  Actual;


280     File1 = fopen (File1Path, "rb");
281     if (!File1)
282     {
283         printf ("Could not open file %s\n", File1Path);
284         return;
285     }

287     Actual = fread (&Header1, 1, sizeof (ACPI_TABLE_HEADER), File1);
288     if (Actual != sizeof (ACPI_TABLE_HEADER))
289     {
290         printf ("File %s does not contain an ACPI table header\n", File1Path);
291         return;
292     }

294     if (!AbValidateHeader (&Header1))
295     {
296         return;
297     }

299     AbPrintHeaderInfo (&Header1);
300 }


303 /******************************************************************************
304  *
305  * FUNCTION:    AbComputeChecksum
306  *
307  * DESCRIPTION: Compute proper checksum for an ACPI table
308  *
309  *****************************************************************************/

311 void
312 AbComputeChecksum (
313     char                    *File1Path)
314 {
315     UINT32                  Actual;
316     ACPI_TABLE_HEADER       *Table;
317     UINT8                   Checksum;


320     File1 = fopen (File1Path, "rb");
321     if (!File1)
322     {
323         printf ("Could not open file %s\n", File1Path);
324         return;
325     }
```



```
327     Actual = fread (&Header1, 1, sizeof (ACPI_TABLE_HEADER), File1);
328     if (Actual < sizeof (ACPI_TABLE_HEADER))
329     {
330         printf ("File %s does not contain an ACPI table header\n", File1Path);
331         return;
332     }

334     if (!AbValidateHeader (&Header1))
335     {
336         return;
337     }

339     if (!Gbl_TerseMode)
340     {
341         AbPrintHeaderInfo (&Header1);
342     }

344     /* Allocate a buffer to hold the entire table */

346     Table = AcpiOsAllocate (Header1.Length);
347     if (!Table)
348     {
349         printf ("could not allocate\n");
350         return;
351     }

353     /* Read the entire table, including header */

355     fseek (File1, 0, SEEK_SET);
356     Actual = fread (Table, 1, Header1.Length, File1);
357     if (Actual != Header1.Length)
358     {
359         printf ("could not read table, length %u\n", Header1.Length);
360         return;
361     }

363     /* Compute the checksum for the table */

365     Table->Checksum = 0;

367     Checksum = (UINT8) (0 - AcpiTbSumTable (Table, Table->Length));
368     printf ("Computed checksum: 0x%X\n\n", Checksum);

370     if (Header1.Checksum == Checksum)
371     {
372         printf ("Checksum ok in AML file, not updating\n");
373         return;
374     }

376     /* Open the target file for writing, to update checksum */

378     fclose (File1);
379     File1 = fopen (File1Path, "r+b");
380     if (!File1)
381     {
382         printf ("Could not open file %s for writing\n", File1Path);
383         return;
384     }

386     /* Set the checksum, write the new header */

388     Header1.Checksum = Checksum;

390     Actual = fwrite (&Header1, 1, sizeof (ACPI_TABLE_HEADER), File1);
391     if (Actual != sizeof (ACPI_TABLE_HEADER))
```

```
392     {
393         printf ("Could not write updated table header\n");
394         return;
395     }

397     printf ("Wrote new checksum\n");
398     return;
399 }


402 /******************************************************************************
403  *
404  * FUNCTION:    AbCompareAmlFiles
405  *
406  * DESCRIPTION: Compare two AML files
407  *
408  *****************************************************************************/

410 int
411 AbCompareAmlFiles (
412     char                    *File1Path,
413     char                    *File2Path)
414 {
415     UINT32                  Actual1;
416     UINT32                  Actual2;
417     UINT32                  Offset;
418     UINT8                   Char1;
419     UINT8                   Char2;
420     UINT8                   Mismatches = 0;
421     BOOLEAN                 HeaderMismatch = FALSE;


424     File1 = fopen (File1Path, "rb");
425     if (!File1)
426     {
427         printf ("Could not open file %s\n", File1Path);
428         return (-1);
429     }

431     File2 = fopen (File2Path, "rb");
432     if (!File2)
433     {
434         printf ("Could not open file %s\n", File2Path);
435         return (-1);
436     }

438     /* Read the ACPI header from each file */

440     Actual1 = fread (&Header1, 1, sizeof (ACPI_TABLE_HEADER), File1);
441     if (Actual1 != sizeof (ACPI_TABLE_HEADER))
442     {
443         printf ("File %s does not contain an ACPI table header\n", File1Path);
444         return (-1);
445     }

447     Actual2 = fread (&Header2, 1, sizeof (ACPI_TABLE_HEADER), File2);
448     if (Actual2 != sizeof (ACPI_TABLE_HEADER))
449     {
450         printf ("File %s does not contain an ACPI table header\n", File2Path);
451         return (-1);
452     }

454     if ((!AbValidateHeader (&Header1)) ||
455         (!AbValidateHeader (&Header2)))
456     {
457         return (-1);
```



```
458     }

460     /* Table signatures must match */

462     if (*((UINT32 *) Header1.Signature) != *((UINT32 *) Header2.Signature))
463     {
464         printf ("Table signatures do not match\n");
465         return (-1);
466     }

468     if (!Gbl_TerseMode)
469     {
470         /* Display header information */

472         AbPrintHeadersInfo (&Header1, &Header2);
473     }

475     if (memcmp (&Header1, &Header2, sizeof (ACPI_TABLE_HEADER)))
476     {
477         printf ("Headers do not match exactly\n");
478         HeaderMismatch = TRUE;
479     }

481     /* Do the byte-by-byte compare */

483     Actual1 = fread (&Char1, 1, 1, File1);
484     Actual2 = fread (&Char2, 1, 1, File2);
485     Offset = sizeof (ACPI_TABLE_HEADER);

487     while ((Actual1 == 1) && (Actual2 == 1))
488     {
489         if (Char1 != Char2)
490         {
491             printf ("Error - Byte mismatch at offset %8.8X: 0x%2.2X 0x%2.2X\n",
492                 Offset, Char1, Char2);
493             Mismatches++;
494             if (Mismatches > 100)
495             {
496                 printf ("100 Mismatches: Too many mismatches\n");
497                 return (-1);
498             }
499         }

501         Offset++;
502         Actual1 = fread (&Char1, 1, 1, File1);
503         Actual2 = fread (&Char2, 1, 1, File2);
504     }

506     if (Actual1)
507     {
508         printf ("Error - file %s is longer than file %s\n", File1Path, File2Path
509         Mismatches++;
510     }
511     else if (Actual2)
512     {
513         printf ("Error - file %s is shorter than file %s\n", File1Path, File2Pat
514         Mismatches++;
515     }
516     else if (!Mismatches)
517     {
518         if (HeaderMismatch)
519         {
520             printf ("Files compare exactly after header\n");
521         }
522         else
523         {
```

```
524             printf ("Files compare exactly\n");
525         }
526     }

528     printf ("%u Mismatches found\n", Mismatches);
529     return (0);
530 }


533 /******************************************************************************
534  *
535  * FUNCTION:    AbGetFileSize
536  *
537  * DESCRIPTION: Get the size of an open file
538  *
539  *****************************************************************************/

541 static UINT32
542 AbGetFileSize (
543     FILE                    *File)
544 {
545     UINT32                  FileSize;
546     long                    Offset;


549     Offset = ftell (File);

551     if (fseek (File, 0, SEEK_END))
552     {
553         return (0);
554     }

556     FileSize = (UINT32) ftell (File);

558     /* Restore file pointer */

560     if (fseek (File, Offset, SEEK_SET))
561     {
562         return (0);
563     }

565     return (FileSize);
566 }


569 /******************************************************************************
570  *
571  * FUNCTION:    AbGetFile
572  *
573  * DESCRIPTION: Open a file and read it entirely into a new buffer
574  *
575  *****************************************************************************/

577 static char *
578 AbGetFile (
579     char                    *Filename,
580     UINT32                  *FileSize)
581 {
582     FILE                    *File;
583     UINT32                  Size;
584     char                    *Buffer = NULL;
585     size_t                  Actual;


588     /* Binary mode does not alter CR/LF pairs */
```



```
590     File = fopen (Filename, "rb");
591     if (!File)
592     {
593         printf ("Could not open file %s\n", Filename);
594         return (NULL);
595     }

597     /* Need file size to allocate a buffer */

599     Size = AbGetFileSize (File);
600     if (!Size)
601     {
602         printf ("Could not get file size (seek) for %s\n", Filename);
603         goto ErrorExit;
604     }

606     /* Allocate a buffer for the entire file */

608     Buffer = calloc (Size, 1);
609     if (!Buffer)
610     {
611         printf ("Could not allocate buffer of size %u\n", Size);
612         goto ErrorExit;
613     }

615     /* Read the entire file */

617     Actual = fread (Buffer, 1, Size, File);
618     if (Actual != Size)
619     {
620         printf ("Could not read the input file %s\n", Filename);
621         free (Buffer);
622         Buffer = NULL;
623         goto ErrorExit;
624     }

626     *FileSize = Size;

628 ErrorExit:
629     fclose (File);
630     return (Buffer);
631 }


634 /******************************************************************************
635  *
636  * FUNCTION:    AbDumpAmlFile
637  *
638  * DESCRIPTION: Dump a binary AML file to a text file
639  *
640  *****************************************************************************/

642 int
643 AbDumpAmlFile (
644     char                    *File1Path,
645     char                    *File2Path)
646 {
647     char                    *FileBuffer;
648     FILE                    *FileOutHandle;
649     UINT32                  FileSize = 0;


652     /* Get the entire AML file, validate header */

654     FileBuffer = AbGetFile (File1Path, &FileSize);
655     if (!FileBuffer)
```

```
656     {
657         return (-1);
658     }

660     printf ("Input file:  %s contains %u (0x%X) bytes\n",
661         File1Path, FileSize, FileSize);

663     FileOutHandle = fopen (File2Path, "wb");
664     if (!FileOutHandle)
665     {
666         printf ("Could not open file %s\n", File2Path);
667         return (-1);
668     }

670     if (!AbValidateHeader ((ACPI_TABLE_HEADER *) FileBuffer))
671     {
672         return (-1);
673     }

675     /* Convert binary AML to text, using common dump buffer routine */

677     AcpiGbl_DebugFile = FileOutHandle;
678     AcpiGbl_DbOutputFlags = ACPI_DB_REDIRECTABLE_OUTPUT;

680     AcpiOsPrintf ("%4.4s @ 0x%8.8X\n",
681         ((ACPI_TABLE_HEADER *) FileBuffer)->Signature, 0);

683     AcpiUtDumpBuffer ((UINT8 *) FileBuffer, FileSize, DB_BYTE_DISPLAY, 0);

685     /* Summary for the output file */

687     FileSize = AbGetFileSize (FileOutHandle);
688     printf ("Output file: %s contains %u (0x%X) bytes\n\n",
689         File2Path, FileSize, FileSize);

691     return (0);
692 }


695 /******************************************************************************
696  *
697  * FUNCTION:    AbExtractAmlFile
698  *
699  * DESCRIPTION: Extract a binary AML file from a text file (as produced by the
700  *              DumpAmlFile procedure or the "acpidump" table utility.
701  *
702  *****************************************************************************/

704 int
705 AbExtractAmlFile (
706     char                    *TableSig,
707     char                    *File1Path,
708     char                    *File2Path)
709 {
710     char                    *Table;
711     char                    Value;
712     UINT32                  i;
713     FILE                    *FileHandle;
714     FILE                    *FileOutHandle;
715     UINT32                  Count = 0;
716     int                     Scanned;


719     /* Open in/out files. input is in text mode, output is in binary mode */

721     FileHandle = fopen (File1Path, "rt");
```



```
722     if (!FileHandle)
723     {
724         printf ("Could not open file %s\n", File1Path);
725         return (-1);
726     }

728     FileOutHandle = fopen (File2Path, "w+b");
729     if (!FileOutHandle)
730     {
731         printf ("Could not open file %s\n", File2Path);
732         return (-1);
733     }

735     /* Force input table sig to uppercase */

737     AcpiUtStrupr (TableSig);


740     /* TBD: examine input for ASCII */


743     /* We have an ascii file, grab one line at a time */

745     while (fgets (Buffer, BUFFER_SIZE, FileHandle))
746     {
747         /* The 4-char ACPI signature appears at the beginning of a line */

749         if (ACPI_COMPARE_NAME (Buffer, TableSig))
750         {
751             printf ("Found table [%4.4s]\n", TableSig);

753             /*
754              * Eat all lines in the table, of the form:
755              *   <offset>: <16 bytes of hex data, separated by spaces> <ASCII re
756              *
757              * Example:
758              *
759              *   02C0: 5F 53 42 5F 4C 4E 4B 44 00 12 13 04 0C FF FF 08  _SB_LNKD
760              *
761              */
762             while (fgets (Buffer, BUFFER_SIZE, FileHandle))
763             {
764                 /* Get past the offset, terminated by a colon */

766                 Table = strchr (Buffer, ':');
767                 if (!Table)
768                 {
769                     /* No colon, all done */
770                     goto Exit;
771                 }

773                 Table += 2; /* Eat the colon + space */

775                 for (i = 0; i < 16; i++)
776                 {
777                     Scanned = AbHexByteToBinary (Table, &Value);
778                     if (!Scanned)
779                     {
780                         goto Exit;
781                     }

783                     Table += 3; /* Go past this hex byte and space */

785                     /* Write the converted (binary) byte */

787                     if (fwrite (&Value, 1, 1, FileOutHandle) != 1)
```

```
788                     {
789                         printf ("Error writing byte %u to output file: %s\n",
790                             Count, File2Path);
791                         goto Exit;
792                     }
793                     Count++;
794                 }
795             }

797             /* No more lines, EOF, all done */

799             goto Exit;
800         }
801     }

803     /* Searched entire file, no match to table signature */

805     printf ("Could not match table signature\n");
806     fclose (FileHandle);
807     return (-1);

809 Exit:
810     printf ("%u (0x%X) bytes written to %s\n", Count, Count, File2Path);
811     fclose (FileHandle);
812     fclose (FileOutHandle);
813     return (0);
814 }


817 /******************************************************************************
818  *
819  * FUNCTION:    Stubs
820  *
821  * DESCRIPTION: For linkage
822  *
823  *****************************************************************************/

825 ACPI_PHYSICAL_ADDRESS
826 AeLocalGetRootPointer (
827     void)
828 {
829     return (AE_OK);
830 }

832 ACPI_THREAD_ID
833 AcpiOsGetThreadId (
834     void)
835 {
836     return (0xFFFF);
837 }

839 ACPI_STATUS
840 AcpiOsExecute (
841     ACPI_EXECUTE_TYPE       Type,
842     ACPI_OSD_EXEC_CALLBACK  Function,
843     void                    *Context)
844 {
845     return (AE_SUPPORT);
846 }
```

```
**********************************************************
    5650 Thu Dec 26 13:50:07 2013
new/usr/src/common/acpica/tools/acpibin/abmain.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: abmain - Main module for the acpi binary utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#define _DECLARE_GLOBALS
#include "acpibin.h"
#include "acapps.h"

/* Local prototypes */

static void
AbDisplayUsage (
    UINT8                   OptionCount);


#define AB_UTILITY_NAME             "ACPI Binary Table Dump Utility"
#define AB_SUPPORTED_OPTIONS        "c:d:e:h:s:tv"


/******************************************************************************
 *
```



```
 * FUNCTION:    AbDisplayUsage
 *
 * DESCRIPTION: Usage message
 *
 *****************************************************************************/

static void
AbDisplayUsage (
    UINT8                   OptionCount)
{

    if (OptionCount)
    {
        printf ("Option requires %u arguments\n\n", OptionCount);
    }

    ACPI_USAGE_HEADER ("acpibin [options]");

    ACPI_OPTION ("-c <File1><File2>",       "Compare two binary AML files");
    ACPI_OPTION ("-d <In><Out>",            "Dump AML binary to text file");
    ACPI_OPTION ("-e <Sig><In><Out>",       "Extract binary AML table from AcpiD
    ACPI_OPTION ("-h <File>",               "Display table header for binary AML
    ACPI_OPTION ("-s <File>",               "Update checksum for binary AML file
    ACPI_OPTION ("-t",                      "Terse mode");
    ACPI_OPTION ("-v",                      "Display version information");
}


/******************************************************************************
 *
 * FUNCTION:    main
 *
 * DESCRIPTION: C main function
 *
 *****************************************************************************/

int ACPI_SYSTEM_XFACE
main (
    int                     argc,
    char                    *argv[])
{
    int                     j;
    int                     Status = AE_OK;


    ACPI_DEBUG_INITIALIZE (); /* For debug version only */

    AcpiGbl_DebugFile = NULL;
    AcpiGbl_DbOutputFlags = DB_CONSOLE_OUTPUT;

    AcpiOsInitialize ();
    printf (ACPI_COMMON_SIGNON (AB_UTILITY_NAME));

    if (argc < 2)
    {
        AbDisplayUsage (0);
        return (0);
    }

    /* Command line options */

    while ((j = AcpiGetopt (argc, argv, AB_SUPPORTED_OPTIONS)) != EOF) switch(j)
    {
    case 'c':   /* Compare Files */

        if (argc < 4)
```

```
128         {
129             AbDisplayUsage (2);
130             return (-1);
131         }

133         Status = AbCompareAmlFiles (AcpiGbl_Optarg, argv[AcpiGbl_Optind]);
134         break;

136     case 'd':   /* Dump AML file */

138         if (argc < 4)
139         {
140             AbDisplayUsage (2);
141             return (-1);
142         }

144         Status = AbDumpAmlFile (AcpiGbl_Optarg, argv[AcpiGbl_Optind]);
145         break;

147     case 'e':   /* Extract AML text file */

149         if (argc < 5)
150         {
151             AbDisplayUsage (3);
152             return (-1);
153         }

155         Status = AbExtractAmlFile (AcpiGbl_Optarg, argv[AcpiGbl_Optind],
156                     argv[AcpiGbl_Optind+1]);
157         break;

159     case 'h':   /* Display ACPI table header */

161         if (argc < 3)
162         {
163             AbDisplayUsage (1);
164             return (-1);
165         }

167         AbDisplayHeader (AcpiGbl_Optarg);
168         return (0);

170     case 's':   /* Compute/update checksum */

172         if (argc < 3)
173         {
174             AbDisplayUsage (1);
175             return (-1);
176         }

178         AbComputeChecksum (AcpiGbl_Optarg);
179         return (0);

181     case 't':   /* Enable terse mode */

183         Gbl_TerseMode = TRUE;
184         break;

186     case 'v': /* -v: (Version): signon already emitted, just exit */

188         return (0);

190     default:

192         AbDisplayUsage (0);
193         return (-1);
```



```
194     }

196     return (Status);
197 }
```

```
**********************************************************
    3116 Thu Dec 26 13:50:07 2013
new/usr/src/common/acpica/tools/acpibin/acpibin.h
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acpibinh - Include file for AcpiBin utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

#include <stdio.h>
#include <fcntl.h>
#include <errno.h>

#define DB_CONSOLE_OUTPUT            0x02
#define ACPI_DB_REDIRECTABLE_OUTPUT  0x01

/*
 * Global variables. Defined in main.c only, externed in all other files
 */
#ifdef _DECLARE_GLOBALS
#define EXTERN
#define INIT_GLOBAL(a,b)        a=b
#else
```



```
#define EXTERN                  extern
#define INIT_GLOBAL(a,b)        a
#endif


/* Globals */

EXTERN BOOLEAN              INIT_GLOBAL (Gbl_TerseMode, FALSE);


/* Prototypes */

int
AbCompareAmlFiles (
    char                    *File1Path,
    char                    *File2Path);

int
AbExtractAmlFile (
    char                    *TableSig,
    char                    *File1Path,
    char                    *File2Path);

int
AbDumpAmlFile (
    char                    *File1Path,
    char                    *File2Path);

void
AbComputeChecksum (
    char                    *File1Path);

void
AbDisplayHeader (
    char                    *File1Path);
```

****
5094 Thu Dec 26 13:50:07 2013
new/usr/src/common/acpica/tools/acpidump/acpidump.h
acpica-unix2-20130823
****

```
/******************************************************************************
 *
 * Module Name: acpidump.h - Include file for AcpiDump utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "actables.h"

#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/stat.h>

/*
 * Global variables. Defined in main.c only, externed in all other files
 */
#ifdef _DECLARE_GLOBALS
#define EXTERN
#define INIT_GLOBAL(a,b)        a=b
#else
#define EXTERN                  extern
#define INIT_GLOBAL(a,b)        a
```



```
#endif


/* Globals */

EXTERN BOOLEAN              INIT_GLOBAL (Gbl_SummaryMode, FALSE);
EXTERN BOOLEAN              INIT_GLOBAL (Gbl_VerboseMode, FALSE);
EXTERN BOOLEAN              INIT_GLOBAL (Gbl_BinaryMode, FALSE);
EXTERN BOOLEAN              INIT_GLOBAL (Gbl_DumpCustomizedTables, FALSE);
EXTERN FILE                 INIT_GLOBAL (*Gbl_OutputFile, NULL);
EXTERN char                 INIT_GLOBAL (*Gbl_OutputFilename, NULL);
EXTERN UINT64               INIT_GLOBAL (Gbl_RsdpBase, 0);

/* Globals required for use with ACPICA modules */

#ifdef _DECLARE_GLOBALS
UINT8                       AcpiGbl_EnableInterpreterSlack = FALSE;
UINT8                       AcpiGbl_IntegerByteWidth = 8;
UINT32                      AcpiDbgLevel = 0;
UINT32                      AcpiDbgLayer = 0;
#endif

/* Action table used to defer requested options */

typedef struct ap_dump_action
{
    char                    *Argument;
    UINT32                  ToBeDone;

} AP_DUMP_ACTION;

/* Local RSDP signature (Not the same as the actual signature which is "RSD PTR

#define AP_DUMP_SIG_RSDP            "RSDP"

#define AP_MAX_ACTIONS              32

#define AP_DUMP_ALL_TABLES          0
#define AP_DUMP_TABLE_BY_ADDRESS    1
#define AP_DUMP_TABLE_BY_NAME       2
#define AP_DUMP_TABLE_BY_FILE       3

#define AP_MAX_ACPI_FILES           256 /* Prevent infinite loops */

/* Minimum FADT sizes for various table addresses */

#define MIN_FADT_FOR_DSDT           (ACPI_FADT_OFFSET (Dsdt) + sizeof (UINT32))
#define MIN_FADT_FOR_FACS           (ACPI_FADT_OFFSET (Facs) + sizeof (UINT32))
#define MIN_FADT_FOR_XDSDT          (ACPI_FADT_OFFSET (XDsdt) + sizeof (UINT64))
#define MIN_FADT_FOR_XFACS          (ACPI_FADT_OFFSET (XFacs) + sizeof (UINT64))


/*
 * apdump - Table get/dump routines
 */
int
ApDumpTableFromFile (
    char                    *Pathname);

int
ApDumpTableByName (
    char                    *Signature);

int
ApDumpTableByAddress (
    char                    *AsciiAddress);
```

```
129 int
130 ApDumpAllTables (
131     void);

133 BOOLEAN
134 ApIsValidHeader (
135     ACPI_TABLE_HEADER       *Table);

137 BOOLEAN
138 ApIsValidChecksum (
139     ACPI_TABLE_HEADER       *Table);

141 UINT32
142 ApGetTableLength (
143     ACPI_TABLE_HEADER       *Table);


146 /*
147  * apfiles - File I/O utilities
148  */
149 UINT32
150 ApGetFileSize (
151     FILE                    *File);

153 int
154 ApOpenOutputFile (
155     char                    *Pathname);

157 int
158 ApWriteToBinaryFile (
159     ACPI_TABLE_HEADER       *Table,
160     UINT32                  Instance);

162 ACPI_TABLE_HEADER *
163 ApGetTableFromFile (
164     char                    *Pathname,
165     UINT32                  *FileSize);
```

```
**********************************************************
   13994 Thu Dec 26 13:50:07 2013
new/usr/src/common/acpica/tools/acpidump/apdump.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: apdump - Dump routines for ACPI tables (acpidump)
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpidump.h"


/* Local prototypes */

static int
ApDumpTableBuffer (
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Instance,
    ACPI_PHYSICAL_ADDRESS   Address);


/******************************************************************************
 *
 * FUNCTION:    ApIsValidHeader
 *
 * PARAMETERS:  Table               - Pointer to table to be validated
```



```
 *
 * RETURN:      TRUE if the header appears to be valid. FALSE otherwise
 *
 * DESCRIPTION: Check for a valid ACPI table header
 *
 *****************************************************************************/

BOOLEAN
ApIsValidHeader (
    ACPI_TABLE_HEADER       *Table)
{
    if (!ACPI_VALIDATE_RSDP_SIG (Table->Signature))
    {
        /* Make sure signature is all ASCII and a valid ACPI name */

        if (!AcpiUtValidAcpiName (Table->Signature))
        {
            fprintf (stderr, "Table signature (0x%8.8X) is invalid\n",
                *(UINT32 *) Table->Signature);
            return (FALSE);
        }

        /* Check for minimum table length */

        if (Table->Length < sizeof (ACPI_TABLE_HEADER))
        {
            fprintf (stderr, "Table length (0x%8.8X) is invalid\n",
                Table->Length);
            return (FALSE);
        }
    }

    return (TRUE);
}


/******************************************************************************
 *
 * FUNCTION:    ApIsValidChecksum
 *
 * PARAMETERS:  Table               - Pointer to table to be validated
 *
 * RETURN:      TRUE if the checksum appears to be valid. FALSE otherwise
 *
 * DESCRIPTION: Check for a valid ACPI table checksum
 *
 *****************************************************************************/

BOOLEAN
ApIsValidChecksum (
    ACPI_TABLE_HEADER       *Table)
{
    ACPI_STATUS             Status;
    ACPI_TABLE_RSDP         *Rsdp;


    if (ACPI_VALIDATE_RSDP_SIG (Table->Signature))
    {
        /*
         * Checksum for RSDP.
         * Note: Other checksums are computed during the table dump.
         */

        Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, Table);
        Status = AcpiTbValidateRsdp (Rsdp);
    }
```

```
127     else
128     {
129         Status = AcpiTbVerifyChecksum (Table, Table->Length);
130     }

132     if (ACPI_FAILURE (Status))
133     {
134         fprintf (stderr, "%4.4s: Warning: wrong checksum\n",
135             Table->Signature);
136     }

138     return (AE_OK);
139 }


142 /******************************************************************************
143  *
144  * FUNCTION:    ApGetTableLength
145  *
146  * PARAMETERS:  Table               - Pointer to the table
147  *
148  * RETURN:      Table length
149  *
150  * DESCRIPTION: Obtain table length according to table signature
151  *
152  *****************************************************************************/

154 UINT32
155 ApGetTableLength (
156     ACPI_TABLE_HEADER       *Table)
157 {
158     ACPI_TABLE_RSDP         *Rsdp;


161     /* Check if table is valid */

163     if (!ApIsValidHeader (Table))
164     {
165         return (0);
166     }

168     if (ACPI_VALIDATE_RSDP_SIG (Table->Signature))
169     {
170         Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, Table);
171         return (Rsdp->Length);
172     }
173     else
174     {
175         return (Table->Length);
176     }
177 }


180 /******************************************************************************
181  *
182  * FUNCTION:    ApDumpTableBuffer
183  *
184  * PARAMETERS:  Table               - ACPI table to be dumped
185  *              Instance            - ACPI table instance no. to be dumped
186  *              Address             - Physical address of the table
187  *
188  * RETURN:      None
189  *
190  * DESCRIPTION: Dump an ACPI table in standard ASCII hex format, with a
191  *              header that is compatible with the AcpiXtract utility.
192  *
```



```
193  *****************************************************************************/

195 static int
196 ApDumpTableBuffer (
197     ACPI_TABLE_HEADER       *Table,
198     UINT32                  Instance,
199     ACPI_PHYSICAL_ADDRESS   Address)
200 {
201     UINT32                  TableLength;


204     TableLength = ApGetTableLength (Table);

206     /* Print only the header if requested */

208     if (Gbl_SummaryMode)
209     {
210         AcpiTbPrintTableHeader (Address, Table);
211         return (0);
212     }

214     /* Dump to binary file if requested */

216     if (Gbl_BinaryMode)
217     {
218         return (ApWriteToBinaryFile (Table, Instance));
219     }

221     /*
222      * Dump the table with header for use with acpixtract utility
223      * Note: simplest to just always emit a 64-bit address. AcpiXtract
224      * utility can handle this.
225      */
226     printf ("%4.4s @ 0x%8.8X%8.8X\n", Table->Signature,
227         ACPI_FORMAT_UINT64 (Address));

229     AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, Table), TableLength,
230         DB_BYTE_DISPLAY, 0);
231     printf ("\n");
232     return (0);
233 }


236 /******************************************************************************
237  *
238  * FUNCTION:    ApDumpAllTables
239  *
240  * PARAMETERS:  None
241  *
242  * RETURN:      Status
243  *
244  * DESCRIPTION: Get all tables from the RSDT/XSDT (or at least all of the
245  *              tables that we can possibly get).
246  *
247  *****************************************************************************/

249 int
250 ApDumpAllTables (
251     void)
252 {
253     ACPI_TABLE_HEADER       *Table;
254     UINT32                  Instance = 0;
255     ACPI_PHYSICAL_ADDRESS   Address;
256     ACPI_STATUS             Status;
257     UINT32                  i;
```

```
260     /* Get and dump all available ACPI tables */

262     for (i = 0; i < AP_MAX_ACPI_FILES; i++)
263     {
264         Status = AcpiOsGetTableByIndex (i, &Table, &Instance, &Address);
265         if (ACPI_FAILURE (Status))
266         {
267             /* AE_LIMIT means that no more tables are available */

269             if (Status == AE_LIMIT)
270             {
271                 return (0);
272             }
273             else if (i == 0)
274             {
275                 fprintf (stderr, "Could not get ACPI tables, %s\n",
276                     AcpiFormatException (Status));
277                 return (-1);
278             }
279             else
280             {
281                 fprintf (stderr, "Could not get ACPI table at index %u, %s\n",
282                     i, AcpiFormatException (Status));
283                 continue;
284             }
285         }

287         if (ApDumpTableBuffer (Table, Instance, Address))
288         {
289             return (-1);
290         }
291         free (Table);
292     }

294     /* Something seriously bad happened if the loop terminates here */

296     return (-1);
297 }


300 /******************************************************************************
301  *
302  * FUNCTION:    ApDumpTableByAddress
303  *
304  * PARAMETERS:  AsciiAddress        - Address for requested ACPI table
305  *
306  * RETURN:      Status
307  *
308  * DESCRIPTION: Get an ACPI table via a physical address and dump it.
309  *
310  *****************************************************************************/

312 int
313 ApDumpTableByAddress (
314     char                    *AsciiAddress)
315 {
316     ACPI_PHYSICAL_ADDRESS   Address;
317     ACPI_TABLE_HEADER       *Table;
318     ACPI_STATUS             Status;
319     int                     TableStatus;
320     UINT64                  LongAddress;


323     /* Convert argument to an integer physical address */
```



```
325     Status = AcpiUtStrtoul64 (AsciiAddress, 0, &LongAddress);
326     if (ACPI_FAILURE (Status))
327     {
328         fprintf (stderr, "%s: Could not convert to a physical address\n",
329             AsciiAddress);
330         return (-1);
331     }

333     Address = (ACPI_PHYSICAL_ADDRESS) LongAddress;
334     Status = AcpiOsGetTableByAddress (Address, &Table);
335     if (ACPI_FAILURE (Status))
336     {
337         fprintf (stderr, "Could not get table at 0x%8.8X%8.8X, %s\n",
338             ACPI_FORMAT_UINT64 (Address),
339             AcpiFormatException (Status));
340         return (-1);
341     }

343     TableStatus = ApDumpTableBuffer (Table, 0, Address);
344     free (Table);
345     return (TableStatus);
346 }


349 /******************************************************************************
350  *
351  * FUNCTION:    ApDumpTableByName
352  *
353  * PARAMETERS:  Signature           - Requested ACPI table signature
354  *
355  * RETURN:      Status
356  *
357  * DESCRIPTION: Get an ACPI table via a signature and dump it. Handles
358  *              multiple tables with the same signature (SSDTs).
359  *
360  *****************************************************************************/

362 int
363 ApDumpTableByName (
364     char                    *Signature)
365 {
366     char                    LocalSignature [ACPI_NAME_SIZE + 1];
367     UINT32                  Instance;
368     ACPI_TABLE_HEADER       *Table;
369     ACPI_PHYSICAL_ADDRESS   Address;
370     ACPI_STATUS             Status;


373     if (strlen (Signature) != ACPI_NAME_SIZE)
374     {
375         fprintf (stderr,
376             "Invalid table signature [%s]: must be exactly 4 characters\n",
377             Signature);
378         return (-1);
379     }

381     /* Table signatures are expected to be uppercase */

383     strcpy (LocalSignature, Signature);
384     AcpiUtStrupr (LocalSignature);

386     /* To be friendly, handle tables whose signatures do not match the name */

388     if (ACPI_COMPARE_NAME (LocalSignature, AP_DUMP_SIG_RSDP))
389     {
390         strcpy (LocalSignature, AP_DUMP_SIG_RSDP);
```

```
391     }
392     else if (ACPI_COMPARE_NAME (LocalSignature, "FADT"))
393     {
394         strcpy (LocalSignature, ACPI_SIG_FADT);
395     }
396     else if (ACPI_COMPARE_NAME (LocalSignature, "MADT"))
397     {
398         strcpy (LocalSignature, ACPI_SIG_MADT);
399     }

401     /* Dump all instances of this signature (to handle multiple SSDTs) */

403     for (Instance = 0; Instance < AP_MAX_ACPI_FILES; Instance++)
404     {
405         Status = AcpiOsGetTableByName (LocalSignature, Instance,
406             &Table, &Address);
407         if (ACPI_FAILURE (Status))
408         {
409             /* AE_LIMIT means that no more tables are available */

411             if (Status == AE_LIMIT)
412             {
413                 return (0);
414             }

416             fprintf (stderr,
417                 "Could not get ACPI table with signature [%s], %s\n",
418                 LocalSignature, AcpiFormatException (Status));
419             return (-1);
420         }

422         if (ApDumpTableBuffer (Table, Instance, Address))
423         {
424             return (-1);
425         }
426         free (Table);
427     }

429     /* Something seriously bad happened if the loop terminates here */

431     return (-1);
432 }


435 /******************************************************************************
436  *
437  * FUNCTION:    ApDumpTableFromFile
438  *
439  * PARAMETERS:  Pathname            - File containing the binary ACPI table
440  *
441  * RETURN:      Status
442  *
443  * DESCRIPTION: Dump an ACPI table from a binary file
444  *
445  *****************************************************************************/

447 int
448 ApDumpTableFromFile (
449     char                    *Pathname)
450 {
451     ACPI_TABLE_HEADER       *Table;
452     UINT32                  FileSize = 0;
453     int                     TableStatus;


456     /* Get the entire ACPI table from the file */
```



```
458     Table = ApGetTableFromFile (Pathname, &FileSize);
459     if (!Table)
460     {
461         return (-1);
462     }

464     /* File must be at least as long as the table length */

466     if (Table->Length > FileSize)
467     {
468         fprintf (stderr,
469             "Table length (0x%X) is too large for input file (0x%X) %s\n",
470             Table->Length, FileSize, Pathname);
471         return (-1);
472     }

474     if (Gbl_VerboseMode)
475     {
476         fprintf (stderr,
477             "Input file:  %s contains table [%4.4s], 0x%X (%u) bytes\n",
478             Pathname, Table->Signature, FileSize, FileSize);
479     }

481     TableStatus = ApDumpTableBuffer (Table, 0, 0);
482     free (Table);
483     return (TableStatus);
484 }


487 /******************************************************************************
488  *
489  * FUNCTION:    AcpiOs* print functions
490  *
491  * DESCRIPTION: Used for linkage with ACPICA modules
492  *
493  *****************************************************************************/

495 void ACPI_INTERNAL_VAR_XFACE
496 AcpiOsPrintf (
497     const char              *Fmt,
498     ...)
499 {
500     va_list                 Args;

502     va_start (Args, Fmt);
503     vfprintf (stdout, Fmt, Args);
504     va_end (Args);
505 }

507 void
508 AcpiOsVprintf (
509     const char              *Fmt,
510     va_list                 Args)
511 {
512     vfprintf (stdout, Fmt, Args);
513 }
```

```
**********************************************************
    8018 Thu Dec 26 13:50:08 2013
new/usr/src/common/acpica/tools/acpidump/apfiles.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: apfiles - File-related functions for acpidump utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpidump.h"
#include "acapps.h"


/******************************************************************************
 *
 * FUNCTION:    ApOpenOutputFile
 *
 * PARAMETERS:  Pathname            - Output filename
 *
 * RETURN:      Open file handle
 *
 * DESCRIPTION: Open a text output file for acpidump. Checks if file already
 *              exists.
 *
 *****************************************************************************/

int
```



```
ApOpenOutputFile (
    char                    *Pathname)
{
    struct stat             StatInfo;
    FILE                    *File;


    /* If file exists, prompt for overwrite */

    if (!stat (Pathname, &StatInfo))
    {
        fprintf (stderr, "Target path already exists, overwrite? [y|n] ");

        if (getchar () != 'y')
        {
            return (-1);
        }
    }

    /* Point stdout to the file */

    File = freopen (Pathname, "w", stdout);
    if (!File)
    {
        perror ("Could not open output file");
        return (-1);
    }

    /* Save the file and path */

    Gbl_OutputFile = File;
    Gbl_OutputFilename = Pathname;
    return (0);
}


/******************************************************************************
 *
 * FUNCTION:    ApWriteToBinaryFile
 *
 * PARAMETERS:  Table               - ACPI table to be written
 *              Instance            - ACPI table instance no. to be written
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Write an ACPI table to a binary file. Builds the output
 *              filename from the table signature.
 *
 *****************************************************************************/

int
ApWriteToBinaryFile (
    ACPI_TABLE_HEADER       *Table,
    UINT32                  Instance)
{
    char                    Filename[ACPI_NAME_SIZE + 16];
    char                    InstanceStr [16];
    FILE                    *File;
    size_t                  Actual;
    UINT32                  TableLength;


    /* Obtain table length */

    TableLength = ApGetTableLength (Table);
```

```
128     /* Construct lower-case filename from the table local signature */

130     if (ACPI_VALIDATE_RSDP_SIG (Table->Signature))
131     {
132         ACPI_MOVE_NAME (Filename, AP_DUMP_SIG_RSDP);
133     }
134     else
135     {
136         ACPI_MOVE_NAME (Filename, Table->Signature);
137     }
138     Filename[0] = (char) ACPI_TOLOWER (Filename[0]);
139     Filename[1] = (char) ACPI_TOLOWER (Filename[1]);
140     Filename[2] = (char) ACPI_TOLOWER (Filename[2]);
141     Filename[3] = (char) ACPI_TOLOWER (Filename[3]);
142     Filename[ACPI_NAME_SIZE] = 0;

144     /* Handle multiple SSDTs - create different filenames for each */

146     if (Instance > 0)
147     {
148         sprintf (InstanceStr, "%u", Instance);
149         strcat (Filename, InstanceStr);
150     }

152     strcat (Filename, ACPI_TABLE_FILE_SUFFIX);

154     if (Gbl_VerboseMode)
155     {
156         fprintf (stderr,
157             "Writing [%4.4s] to binary file: %s 0x%X (%u) bytes\n",
158             Table->Signature, Filename, Table->Length, Table->Length);
159     }

161     /* Open the file and dump the entire table in binary mode */

163     File = fopen (Filename, "wb");
164     if (!File)
165     {
166         perror ("Could not open output file");
167         return (-1);
168     }

170     Actual = fwrite (Table, 1, TableLength, File);
171     if (Actual != TableLength)
172     {
173         perror ("Error writing binary output file");
174         fclose (File);
175         return (-1);
176     }

178     fclose (File);
179     return (0);
180 }


183 /******************************************************************************
184  *
185  * FUNCTION:    ApGetTableFromFile
186  *
187  * PARAMETERS:  Pathname            - File containing the binary ACPI table
188  *              OutFileSize         - Where the file size is returned
189  *
190  * RETURN:      Buffer containing the ACPI table. NULL on error.
191  *
192  * DESCRIPTION: Open a file and read it entirely into a new buffer
193  *
```



```
194  *****************************************************************************/

196 ACPI_TABLE_HEADER *
197 ApGetTableFromFile (
198     char                    *Pathname,
199     UINT32                  *OutFileSize)
200 {
201     ACPI_TABLE_HEADER       *Buffer = NULL;
202     FILE                    *File;
203     UINT32                  FileSize;
204     size_t                  Actual;


207     /* Must use binary mode */

209     File = fopen (Pathname, "rb");
210     if (!File)
211     {
212         perror ("Could not open input file");
213         return (NULL);
214     }

216     /* Need file size to allocate a buffer */

218     FileSize = ApGetFileSize (File);
219     if (!FileSize)
220     {
221         fprintf (stderr,
222             "Could not get input file size: %s\n", Pathname);
223         goto Cleanup;
224     }

226     /* Allocate a buffer for the entire file */

228     Buffer = calloc (1, FileSize);
229     if (!Buffer)
230     {
231         fprintf (stderr,
232             "Could not allocate file buffer of size: %u\n", FileSize);
233         goto Cleanup;
234     }

236     /* Read the entire file */

238     Actual = fread (Buffer, 1, FileSize, File);
239     if (Actual != FileSize)
240     {
241         fprintf (stderr,
242             "Could not read input file: %s\n", Pathname);
243         free (Buffer);
244         Buffer = NULL;
245         goto Cleanup;
246     }

248     *OutFileSize = FileSize;

250 Cleanup:
251     fclose (File);
252     return (Buffer);
253 }


256 /******************************************************************************
257  *
258  * FUNCTION:    ApGetFileSize
259  *
```

```
 * PARAMETERS:  File                - Open file descriptor
 *
 * RETURN:      File size in bytes
 *
 * DESCRIPTION: Get the size of an open file
 *
 *****************************************************************************/

UINT32
ApGetFileSize (
    FILE                    *File)
{
    UINT32                  FileSize;
    long                    Offset;


    Offset = ftell (File);
    if (fseek (File, 0, SEEK_END))
    {
        return (0);
    }

    /* Get size and restore file pointer */

    FileSize = (UINT32) ftell (File);
    if (fseek (File, Offset, SEEK_SET))
    {
        return (0);
    }

    return (FileSize);
}
```

```
**********************************************************
   10358 Thu Dec 26 13:50:08 2013
new/usr/src/common/acpica/tools/acpidump/apmain.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: apmain - Main module for the acpidump utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define _DECLARE_GLOBALS
#include "acpidump.h"
#include "acapps.h"


/*
 * acpidump - A portable utility for obtaining system ACPI tables and dumping
 * them in an ASCII hex format suitable for binary extraction via acpixtract.
 *
 * Obtaining the system ACPI tables is an OS-specific operation.
 *
 * This utility can be ported to any host operating system by providing a
 * module containing system-specific versions of these interfaces:
 *
 *      AcpiOsGetTableByAddress
 *      AcpiOsGetTableByIndex
 *      AcpiOsGetTableByName
 *
```



```
 * See the ACPICA Reference Guide for the exact definitions of these
 * interfaces. Also, see these ACPICA source code modules for example
 * implementations:
 *
 *      source/os_specific/service_layers/oswintbl.c
 *      source/os_specific/service_layers/oslinuxtbl.c
 */


/* Local prototypes */

static void
ApDisplayUsage (
    void);

static int
ApDoOptions (
    int                     argc,
    char                    **argv);

static void
ApInsertAction (
    char                    *Argument,
    UINT32                  ToBeDone);


/* Table for deferred actions from command line options */

AP_DUMP_ACTION              ActionTable [AP_MAX_ACTIONS];
UINT32                      CurrentAction = 0;


#define AP_UTILITY_NAME             "ACPI Binary Table Dump Utility"
#define AP_SUPPORTED_OPTIONS        "?a:bcf:hn:o:r:svz"


/******************************************************************************
 *
 * FUNCTION:    ApDisplayUsage
 *
 * DESCRIPTION: Usage message for the AcpiDump utility
 *
 *****************************************************************************/

static void
ApDisplayUsage (
    void)
{

    ACPI_USAGE_HEADER ("acpidump [options]");

    ACPI_OPTION ("-b",                      "Dump tables to binary files");
    ACPI_OPTION ("-c",                      "Dump customized tables");
    ACPI_OPTION ("-h -?",                   "This help message");
    ACPI_OPTION ("-o <File>",               "Redirect output to file");
    ACPI_OPTION ("-r <Address>",            "Dump tables from specified RSDP");
    ACPI_OPTION ("-s",                      "Print table summaries only");
    ACPI_OPTION ("-v",                      "Display version information");
    ACPI_OPTION ("-z",                      "Verbose mode");

    printf ("\nTable Options:\n");

    ACPI_OPTION ("-a <Address>",            "Get table via a physical address");
    ACPI_OPTION ("-f <BinaryFile>",         "Get table via a binary file");
    ACPI_OPTION ("-n <Signature>",          "Get table via a name/signature");
```

```
128     printf (
129         "\n"
130         "Invocation without parameters dumps all available tables\n"
131         "Multiple mixed instances of -a, -f, and -n are supported\n\n");
132 }


135 /******************************************************************************
136  *
137  * FUNCTION:    ApInsertAction
138  *
139  * PARAMETERS:  Argument            - Pointer to the argument for this action
140  *              ToBeDone            - What to do to process this action
141  *
142  * RETURN:      None. Exits program if action table becomes full.
143  *
144  * DESCRIPTION: Add an action item to the action table
145  *
146  *****************************************************************************/

148 static void
149 ApInsertAction (
150     char                    *Argument,
151     UINT32                  ToBeDone)
152 {

154     /* Insert action and check for table overflow */

156     ActionTable [CurrentAction].Argument = Argument;
157     ActionTable [CurrentAction].ToBeDone = ToBeDone;

159     CurrentAction++;
160     if (CurrentAction > AP_MAX_ACTIONS)
161     {
162         fprintf (stderr, "Too many table options (max %u)\n", AP_MAX_ACTIONS);
163         exit (-1);
164     }
165 }


168 /******************************************************************************
169  *
170  * FUNCTION:    ApDoOptions
171  *
172  * PARAMETERS:  argc/argv           - Standard argc/argv
173  *
174  * RETURN:      Status
175  *
176  * DESCRIPTION: Command line option processing. The main actions for getting
177  *              and dumping tables are deferred via the action table.
178  *
179  *****************************************************************************/

181 static int
182 ApDoOptions (
183     int                     argc,
184     char                    **argv)
185 {
186     int                     j;
187     ACPI_STATUS             Status;


190     /* Command line options */

192     while ((j = AcpiGetopt (argc, argv, AP_SUPPORTED_OPTIONS)) != EOF) switch (j
193     {
```



```
194     /*
195      * Global options
196      */
197     case 'b':   /* Dump all input tables to binary files */

199         Gbl_BinaryMode = TRUE;
200         continue;

202     case 'c':   /* Dump customized tables */

204         Gbl_DumpCustomizedTables = TRUE;
205         continue;

207     case 'h':
208     case '?':

210         ApDisplayUsage ();
211         exit (0);

213     case 'o':   /* Redirect output to a single file */

215         if (ApOpenOutputFile (AcpiGbl_Optarg))
216         {
217             exit (-1);
218         }
219         continue;

221     case 'r':   /* Dump tables from specified RSDP */

223         Status = AcpiUtStrtoul64 (AcpiGbl_Optarg, 0, &Gbl_RsdpBase);
224         if (ACPI_FAILURE (Status))
225         {
226             fprintf (stderr, "%s: Could not convert to a physical address\n",
227                 AcpiGbl_Optarg);
228             exit (-1);
229         }
230         continue;

232     case 's':   /* Print table summaries only */

234         Gbl_SummaryMode = TRUE;
235         continue;

237     case 'v':   /* Revision/version */

239         printf (ACPI_COMMON_SIGNON (AP_UTILITY_NAME));
240         exit (0);

242     case 'z':   /* Verbose mode */

244         Gbl_VerboseMode = TRUE;
245         fprintf (stderr, ACPI_COMMON_SIGNON (AP_UTILITY_NAME));
246         continue;

248     /*
249      * Table options
250      */
251     case 'a':   /* Get table by physical address */

253         ApInsertAction (AcpiGbl_Optarg, AP_DUMP_TABLE_BY_ADDRESS);
254         break;

256     case 'f':   /* Get table from a file */

258         ApInsertAction (AcpiGbl_Optarg, AP_DUMP_TABLE_BY_FILE);
259         break;
```

```
261     case 'n':   /* Get table by input name (signature) */

263         ApInsertAction (AcpiGbl_Optarg, AP_DUMP_TABLE_BY_NAME);
264         break;

266     default:

268         ApDisplayUsage ();
269         exit (-1);
270     }

272     /* If there are no actions, this means "get/dump all tables" */

274     if (CurrentAction == 0)
275     {
276         ApInsertAction (NULL, AP_DUMP_ALL_TABLES);
277     }

279     return (0);
280 }


283 /******************************************************************************
284  *
285  * FUNCTION:    main
286  *
287  * PARAMETERS:  argc/argv           - Standard argc/argv
288  *
289  * RETURN:      Status
290  *
291  * DESCRIPTION: C main function for acpidump utility
292  *
293  *****************************************************************************/

295 int ACPI_SYSTEM_XFACE
296 main (
297     int                     argc,
298     char                    *argv[])
299 {
300     int                     Status = 0;
301     AP_DUMP_ACTION          *Action;
302     UINT32                  FileSize;
303     UINT32                  i;


306     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

308     /* Process command line options */

310     if (ApDoOptions (argc, argv))
311     {
312         return (-1);
313     }

315     /* Get/dump ACPI table(s) as requested */

317     for (i = 0; i < CurrentAction; i++)
318     {
319         Action = &ActionTable[i];
320         switch (Action->ToBeDone)
321         {
322         case AP_DUMP_ALL_TABLES:

324             Status = ApDumpAllTables ();
325             break;
```



```
327         case AP_DUMP_TABLE_BY_ADDRESS:

329             Status = ApDumpTableByAddress (Action->Argument);
330             break;

332         case AP_DUMP_TABLE_BY_NAME:

334             Status = ApDumpTableByName (Action->Argument);
335             break;

337         case AP_DUMP_TABLE_BY_FILE:

339             Status = ApDumpTableFromFile (Action->Argument);
340             break;

342         default:

344             fprintf (stderr, "Internal error, invalid action: 0x%X\n",
345                 Action->ToBeDone);
346             return (-1);
347         }

349         if (Status)
350         {
351             return (Status);
352         }
353     }

355     if (Gbl_OutputFile)
356     {
357         if (Gbl_VerboseMode)
358         {
359             /* Summary for the output file */

361             FileSize = ApGetFileSize (Gbl_OutputFile);
362             fprintf (stderr, "Output file %s contains 0x%X (%u) bytes\n\n",
363                 Gbl_OutputFilename, FileSize, FileSize);
364         }

366         fclose (Gbl_OutputFile);
367     }

369     return (Status);
370 }
```

```
**********************************************************
    5432 Thu Dec 26 13:50:08 2013
new/usr/src/common/acpica/tools/acpiexec/aecommon.h
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: aecommon - common include for the AcpiExec utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _AECOMMON
#define _AECOMMON

#ifdef _MSC_VER                 /* disable some level-4 warnings */
#pragma warning(disable:4100)   /* warning C4100: unreferenced formal parameter
#endif

#include "acpi.h"
#include "accommon.h"
#include "acparser.h"
#include "amlcode.h"
#include "acnamesp.h"
#include "acdebug.h"
#include "actables.h"
#include "acinterp.h"
#include "acapps.h"
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>

extern BOOLEAN              AcpiGbl_IgnoreErrors;
extern UINT8                AcpiGbl_RegionFillValue;
extern UINT8                AcpiGbl_UseHwReducedFadt;
extern BOOLEAN              AcpiGbl_DisplayRegionAccess;
extern BOOLEAN              AcpiGbl_DoInterfaceTests;

/* Check for unexpected exceptions */

#define AE_CHECK_STATUS(Name, Status, Expected) \
    if (Status != Expected) \
    { \
        AcpiOsPrintf ("Unexpected %s from %s (%s-%d)\n", \
            AcpiFormatException (Status), #Name, _AcpiModuleName, __LINE__); \
    }

/* Check for unexpected non-AE_OK errors */

#define AE_CHECK_OK(Name, Status)   AE_CHECK_STATUS (Name, Status, AE_OK);

typedef struct ae_table_desc
{
    ACPI_TABLE_HEADER       *Table;
    struct ae_table_desc    *Next;

} AE_TABLE_DESC;

/*
 * Debug Regions
 */
typedef struct ae_region
{
    ACPI_PHYSICAL_ADDRESS   Address;
    UINT32                  Length;
    void                    *Buffer;
    void                    *NextRegion;
    UINT8                   SpaceId;

} AE_REGION;

typedef struct ae_debug_regions
{
    UINT32                  NumberOfRegions;
    AE_REGION               *RegionList;

} AE_DEBUG_REGIONS;


#define TEST_OUTPUT_LEVEL(lvl)          if ((lvl) & OutputLevel)

#define OSD_PRINT(lvl,fp)               TEST_OUTPUT_LEVEL(lvl) {\
                                            AcpiOsPrintf PARAM_LIST(fp);}

void ACPI_SYSTEM_XFACE
AeCtrlCHandler (
    int                     Sig);

ACPI_STATUS
AeBuildLocalTables (
    UINT32                  TableCount,
    AE_TABLE_DESC           *TableList);
```

```
127 ACPI_STATUS
128 AeInstallTables (
129     void);

131 void
132 AeDumpNamespace (
133     void);

135 void
136 AeDumpObject (
137     char                    *MethodName,
138     ACPI_BUFFER             *ReturnObj);

140 void
141 AeDumpBuffer (
142     UINT32                  Address);

144 void
145 AeExecute (
146     char                    *Name);

148 void
149 AeSetScope (
150     char                    *Name);

152 void
153 AeCloseDebugFile (
154     void);

156 void
157 AeOpenDebugFile (
158     char                    *Name);

160 ACPI_STATUS
161 AeDisplayAllMethods (
162     UINT32                  DisplayCount);

164 ACPI_STATUS
165 AeInstallEarlyHandlers (
166     void);

168 ACPI_STATUS
169 AeInstallLateHandlers (
170     void);

172 void
173 AeMiscellaneousTests (
174     void);

176 ACPI_STATUS
177 AeRegionHandler (
178     UINT32                  Function,
179     ACPI_PHYSICAL_ADDRESS   Address,
180     UINT32                  BitWidth,
181     UINT64                  *Value,
182     void                    *HandlerContext,
183     void                    *RegionContext);

185 UINT32
186 AeGpeHandler (
187     ACPI_HANDLE             GpeDevice,
188     UINT32                  GpeNumber,
189     void                    *Context);

191 void
```



```
192 AeGlobalEventHandler (
193     UINT32                  Type,
194     ACPI_HANDLE             GpeDevice,
195     UINT32                  EventNumber,
196     void                    *Context);

198 #endif /* _AECOMMON */
```

```
**********************************************************
   22310 Thu Dec 26 13:50:08 2013
new/usr/src/common/acpica/tools/acpiexec/aeexec.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aeexec - Support routines for AcpiExec utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aecommon.h"

#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("aeexec")

/* Local prototypes */

static ACPI_STATUS
AeSetupConfiguration (
    void                    *RegionAddr);

static void
AeTestBufferArgument (
    void);

static void
```



```
AeTestPackageArgument (
    void);

static ACPI_STATUS
AeGetDevices (
    ACPI_HANDLE             ObjHandle,
    UINT32                  NestingLevel,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_STATUS
ExecuteOSI (
    char                    *OsiString,
    UINT32                  ExpectedResult);

static void
AeMutexInterfaces (
    void);

static void
AeHardwareInterfaces (
    void);

static void
AeGenericRegisters (
    void);

#if (!ACPI_REDUCED_HARDWARE)
static void
AfInstallGpeBlock (
    void);
#endif /* !ACPI_REDUCED_HARDWARE */

extern unsigned char Ssdt2Code[];
extern unsigned char Ssdt3Code[];
extern unsigned char Ssdt4Code[];


/******************************************************************************
 *
 * FUNCTION:    AeSetupConfiguration
 *
 * PARAMETERS:  RegionAddr          - Address for an ACPI table to be loaded
 *                                    dynamically. Test purposes only.
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Call AML _CFG configuration control method
 *
 *****************************************************************************/

static ACPI_STATUS
AeSetupConfiguration (
    void                    *RegionAddr)
{
    ACPI_OBJECT_LIST        ArgList;
    ACPI_OBJECT             Arg[3];


    /*
     * Invoke _CFG method if present
     */
    ArgList.Count = 1;
    ArgList.Pointer = Arg;

    Arg[0].Type = ACPI_TYPE_INTEGER;
```

```
126     Arg[0].Integer.Value = ACPI_TO_INTEGER (RegionAddr);
127
128     (void) AcpiEvaluateObject (NULL, "\\_CFG", &ArgList, NULL);
129     return (AE_OK);
130 }
131
132
133 #if (!ACPI_REDUCED_HARDWARE)
134 /*******************************************************************************
135  *
136  * FUNCTION:    AfInstallGpeBlock
137  *
138  * PARAMETERS:  None
139  *
140  * RETURN:      None
141  *
142  * DESCRIPTION: Test GPE block device initialization. Requires test ASL with
143  *              A \GPE2 device.
144  *
145  ******************************************************************************/
146
147 static void
148 AfInstallGpeBlock (
149     void)
150 {
151     ACPI_STATUS                 Status;
152     ACPI_HANDLE                 Handle;
153     ACPI_HANDLE                 Handle2 = NULL;
154     ACPI_HANDLE                 Handle3 = NULL;
155     ACPI_GENERIC_ADDRESS        BlockAddress;
156     ACPI_HANDLE                 GpeDevice;
157
158
159     Status = AcpiGetHandle (NULL, "\\_GPE", &Handle);
160     if (ACPI_FAILURE (Status))
161     {
162         return;
163     }
164
165     ACPI_MEMSET (&BlockAddress, 0, sizeof (ACPI_GENERIC_ADDRESS));
166     BlockAddress.SpaceId = ACPI_ADR_SPACE_SYSTEM_MEMORY;
167     BlockAddress.Address = 0x76540000;
168
169     Status = AcpiGetHandle (NULL, "\\GPE2", &Handle2);
170     if (ACPI_SUCCESS (Status))
171     {
172         Status = AcpiInstallGpeBlock (Handle2, &BlockAddress, 7, 8);
173         AE_CHECK_OK (AcpiInstallGpeBlock, Status);
174
175         Status = AcpiInstallGpeHandler (Handle2, 8,
176             ACPI_GPE_LEVEL_TRIGGERED, AeGpeHandler, NULL);
177         AE_CHECK_OK (AcpiInstallGpeHandler, Status);
178
179         Status = AcpiEnableGpe (Handle2, 8);
180         AE_CHECK_OK (AcpiEnableGpe, Status);
181
182         Status = AcpiGetGpeDevice (0x30, &GpeDevice);
183         AE_CHECK_OK (AcpiGetGpeDevice, Status);
184
185         Status = AcpiGetGpeDevice (0x42, &GpeDevice);
186         AE_CHECK_OK (AcpiGetGpeDevice, Status);
187
188         Status = AcpiGetGpeDevice (AcpiCurrentGpeCount-1, &GpeDevice);
189         AE_CHECK_OK (AcpiGetGpeDevice, Status);
190
191         Status = AcpiGetGpeDevice (AcpiCurrentGpeCount, &GpeDevice);
```



```
192         AE_CHECK_STATUS (AcpiGetGpeDevice, Status, AE_NOT_EXIST);
193
194         Status = AcpiRemoveGpeHandler (Handle2, 8, AeGpeHandler);
195         AE_CHECK_OK (AcpiRemoveGpeHandler, Status);
196     }
197
198     Status = AcpiGetHandle (NULL, "\\GPE3", &Handle3);
199     if (ACPI_SUCCESS (Status))
200     {
201         Status = AcpiInstallGpeBlock (Handle3, &BlockAddress, 8, 11);
202         AE_CHECK_OK (AcpiInstallGpeBlock, Status);
203     }
204 }
205 #endif /* !ACPI_REDUCED_HARDWARE */
206
207
208 /* Test using a Buffer object as a method argument */
209
210 static void
211 AeTestBufferArgument (
212     void)
213 {
214     ACPI_OBJECT_LIST        Params;
215     ACPI_OBJECT             BufArg;
216     UINT8                   Buffer[] = {
217         0,0,0,0,
218         4,0,0,0,
219         1,2,3,4};
220
221
222     BufArg.Type = ACPI_TYPE_BUFFER;
223     BufArg.Buffer.Length = 12;
224     BufArg.Buffer.Pointer = Buffer;
225
226     Params.Count = 1;
227     Params.Pointer = &BufArg;
228
229     (void) AcpiEvaluateObject (NULL, "\\BUF", &Params, NULL);
230 }
231
232
233 static ACPI_OBJECT                  PkgArg;
234 static ACPI_OBJECT                  PkgElements[5];
235 static ACPI_OBJECT                  Pkg2Elements[5];
236 static ACPI_OBJECT_LIST             Params;
237
238
239 /*
240  * Test using a Package object as an method argument
241  */
242 static void
243 AeTestPackageArgument (
244     void)
245 {
246
247     /* Main package */
248
249     PkgArg.Type = ACPI_TYPE_PACKAGE;
250     PkgArg.Package.Count = 4;
251     PkgArg.Package.Elements = PkgElements;
252
253     /* Main package elements */
254
255     PkgElements[0].Type = ACPI_TYPE_INTEGER;
256     PkgElements[0].Integer.Value = 0x22228888;
```

```
258     PkgElements[1].Type = ACPI_TYPE_STRING;
259     PkgElements[1].String.Length = sizeof ("Top-level package");
260     PkgElements[1].String.Pointer = "Top-level package";

262     PkgElements[2].Type = ACPI_TYPE_BUFFER;
263     PkgElements[2].Buffer.Length = sizeof ("XXXX");
264     PkgElements[2].Buffer.Pointer = (UINT8 *) "XXXX";

266     PkgElements[3].Type = ACPI_TYPE_PACKAGE;
267     PkgElements[3].Package.Count = 2;
268     PkgElements[3].Package.Elements = Pkg2Elements;

270     /* Sub-package elements */

272     Pkg2Elements[0].Type = ACPI_TYPE_INTEGER;
273     Pkg2Elements[0].Integer.Value = 0xAAAABBBB;

275     Pkg2Elements[1].Type = ACPI_TYPE_STRING;
276     Pkg2Elements[1].String.Length = sizeof ("Nested Package");
277     Pkg2Elements[1].String.Pointer = "Nested Package";

279     /* Parameter object */

281     Params.Count = 1;
282     Params.Pointer = &PkgArg;

284     (void) AcpiEvaluateObject (NULL, "\\_PKG", &Params, NULL);
285 }


288 static ACPI_STATUS
289 AeGetDevices (
290     ACPI_HANDLE                     ObjHandle,
291     UINT32                          NestingLevel,
292     void                            *Context,
293     void                            **ReturnValue)
294 {

296     return (AE_OK);
297 }


300 /*****************************************************************************
301  *
302  * FUNCTION:    ExecuteOSI
303  *
304  * PARAMETERS:  OsiString           - String passed to _OSI method
305  *              ExpectedResult      - 0 (FALSE) or 0xFFFFFFFF (TRUE)
306  *
307  * RETURN:      Status
308  *
309  * DESCRIPTION: Execute the internally implemented (in ACPICA) _OSI method.
310  *
311  *****************************************************************************/

313 static ACPI_STATUS
314 ExecuteOSI (
315     char                    *OsiString,
316     UINT32                  ExpectedResult)
317 {
318     ACPI_STATUS             Status;
319     ACPI_OBJECT_LIST        ArgList;
320     ACPI_OBJECT             Arg[1];
321     ACPI_BUFFER             ReturnValue;
322     ACPI_OBJECT             *Obj;
```



```
325     /* Setup input argument */

327     ArgList.Count = 1;
328     ArgList.Pointer = Arg;

330     Arg[0].Type = ACPI_TYPE_STRING;
331     Arg[0].String.Pointer = OsiString;
332     Arg[0].String.Length = strlen (Arg[0].String.Pointer);

334     /* Ask ACPICA to allocate space for the return object */

336     ReturnValue.Length = ACPI_ALLOCATE_BUFFER;

338     Status = AcpiEvaluateObject (NULL, "\\_OSI", &ArgList, &ReturnValue);

340     if (ACPI_FAILURE (Status))
341     {
342         AcpiOsPrintf ("Could not execute _OSI method, %s\n",
343             AcpiFormatException (Status));
344         return (Status);
345     }

347     Status = AE_ERROR;

349     if (ReturnValue.Length < sizeof (ACPI_OBJECT))
350     {
351         AcpiOsPrintf ("Return value from _OSI method too small, %.8X\n",
352             ReturnValue.Length);
353         goto ErrorExit;
354     }

356     Obj = ReturnValue.Pointer;
357     if (Obj->Type != ACPI_TYPE_INTEGER)
358     {
359         AcpiOsPrintf ("Invalid return type from _OSI method, %.2X\n", Obj->Type)
360         goto ErrorExit;
361     }

363     if (Obj->Integer.Value != ExpectedResult)
364     {
365         AcpiOsPrintf ("Invalid return value from _OSI, expected %.8X found %.8X\
366             ExpectedResult, (UINT32) Obj->Integer.Value);
367         goto ErrorExit;
368     }

370     Status = AE_OK;

372     /* Reset the OSI data */

374     AcpiGbl_OsiData = 0;

376 ErrorExit:

378     /* Free a buffer created via ACPI_ALLOCATE_BUFFER */

380     AcpiOsFree (ReturnValue.Pointer);

382     return (Status);
383 }


386 /*****************************************************************************
387  *
388  * FUNCTION:    AeGenericRegisters
389  *
```

```
 * DESCRIPTION: Call the AcpiRead/Write interfaces.
 *
 *****************************************************************************/

static ACPI_GENERIC_ADDRESS       GenericRegister;

static void
AeGenericRegisters (
    void)
{
    ACPI_STATUS             Status;
    UINT64                  Value;


    GenericRegister.Address = 0x1234;
    GenericRegister.BitWidth = 64;
    GenericRegister.BitOffset = 0;
    GenericRegister.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;

    Status = AcpiRead (&Value, &GenericRegister);
    AE_CHECK_OK (AcpiRead, Status);

    Status = AcpiWrite (Value, &GenericRegister);
    AE_CHECK_OK (AcpiWrite, Status);

    GenericRegister.Address = 0x12345678;
    GenericRegister.BitOffset = 0;
    GenericRegister.SpaceId = ACPI_ADR_SPACE_SYSTEM_MEMORY;

    Status = AcpiRead (&Value, &GenericRegister);
    AE_CHECK_OK (AcpiRead, Status);

    Status = AcpiWrite (Value, &GenericRegister);
    AE_CHECK_OK (AcpiWrite, Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AeMutexInterfaces
 *
 * DESCRIPTION: Exercise the AML mutex access interfaces
 *
 *****************************************************************************/

static void
AeMutexInterfaces (
    void)
{
    ACPI_STATUS             Status;
    ACPI_HANDLE             MutexHandle;


    /* Get a handle to an AML mutex */

    Status = AcpiGetHandle (NULL, "\\MTX1", &MutexHandle);
    if (Status == AE_NOT_FOUND)
    {
        return;
    }

    AE_CHECK_OK (AcpiGetHandle, Status);
    if (ACPI_FAILURE (Status))
    {
        return;
    }
```



```
    /* Acquire the  mutex */

    Status = AcpiAcquireMutex (NULL, "\\MTX1", 0xFFFF);
    AE_CHECK_OK (AcpiAcquireMutex, Status);
    if (ACPI_FAILURE (Status))
    {
        return;
    }

    /* Release mutex with different parameters */

    Status = AcpiReleaseMutex (MutexHandle, NULL);
    AE_CHECK_OK (AcpiReleaseMutex, Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AeHardwareInterfaces
 *
 * DESCRIPTION: Call various hardware support interfaces
 *
 *****************************************************************************/

static void
AeHardwareInterfaces (
    void)
{
#if (!ACPI_REDUCED_HARDWARE)

    ACPI_STATUS             Status;
    UINT32                  Value;


    /* If Hardware Reduced flag is set, we are all done */

    if (AcpiGbl_ReducedHardware)
    {
        return;
    }

    Status = AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, 1);
    AE_CHECK_OK (AcpiWriteBitRegister, Status);

    Status = AcpiWriteBitRegister (ACPI_BITREG_GLOBAL_LOCK_ENABLE, 1);
    AE_CHECK_OK (AcpiWriteBitRegister, Status);

    Status = AcpiWriteBitRegister (ACPI_BITREG_SLEEP_ENABLE, 1);
    AE_CHECK_OK (AcpiWriteBitRegister, Status);

    Status = AcpiWriteBitRegister (ACPI_BITREG_ARB_DISABLE, 1);
    AE_CHECK_OK (AcpiWriteBitRegister, Status);


    Status = AcpiReadBitRegister (ACPI_BITREG_WAKE_STATUS, &Value);
    AE_CHECK_OK (AcpiReadBitRegister, Status);

    Status = AcpiReadBitRegister (ACPI_BITREG_GLOBAL_LOCK_ENABLE, &Value);
    AE_CHECK_OK (AcpiReadBitRegister, Status);

    Status = AcpiReadBitRegister (ACPI_BITREG_SLEEP_ENABLE, &Value);
    AE_CHECK_OK (AcpiReadBitRegister, Status);

    Status = AcpiReadBitRegister (ACPI_BITREG_ARB_DISABLE, &Value);
    AE_CHECK_OK (AcpiReadBitRegister, Status);
```

```
#endif /* !ACPI_REDUCED_HARDWARE */
}


/******************************************************************************
 *
 * FUNCTION:    AeMiscellaneousTests
 *
 * DESCRIPTION: Various ACPICA validation tests.
 *
 *****************************************************************************/

void
AeMiscellaneousTests (
    void)
{
    ACPI_BUFFER             ReturnBuf;
    char                    Buffer[32];
    ACPI_STATUS             Status;
    ACPI_STATISTICS         Stats;
    ACPI_HANDLE             Handle;

#if (!ACPI_REDUCED_HARDWARE)
    ACPI_VENDOR_UUID        Uuid = {0, {ACPI_INIT_UUID (0,0,0,0,0,0,0,0,0,0,0)}};
    UINT32                  LockHandle1;
    UINT32                  LockHandle2;
#endif /* !ACPI_REDUCED_HARDWARE */


    Status = AcpiGetHandle (NULL, "\\", &Handle);
    AE_CHECK_OK (AcpiGetHandle, Status);

    if (AcpiGbl_DoInterfaceTests)
    {
        /*
         * Tests for AcpiLoadTable and AcpiUnloadParentTable
         */

        /* Attempt unload of DSDT, should fail */

        Status = AcpiGetHandle (NULL, "\\_SB_", &Handle);
        AE_CHECK_OK (AcpiGetHandle, Status);

        Status = AcpiUnloadParentTable (Handle);
        AE_CHECK_STATUS (AcpiUnloadParentTable, Status, AE_TYPE);

        /* Load and unload SSDT4 */

        Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt4Code);
        AE_CHECK_OK (AcpiLoadTable, Status);

        Status = AcpiGetHandle (NULL, "\\_T96", &Handle);
        AE_CHECK_OK (AcpiGetHandle, Status);

        Status = AcpiUnloadParentTable (Handle);
        AE_CHECK_OK (AcpiUnloadParentTable, Status);

        /* Re-load SSDT4 */

        Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt4Code);
        AE_CHECK_OK (AcpiLoadTable, Status);

        /* Unload and re-load SSDT2 (SSDT2 is in the XSDT) */

        Status = AcpiGetHandle (NULL, "\\_T99", &Handle);
```



```
        AE_CHECK_OK (AcpiGetHandle, Status);

        Status = AcpiUnloadParentTable (Handle);
        AE_CHECK_OK (AcpiUnloadParentTable, Status);

        Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt2Code);
        AE_CHECK_OK (AcpiLoadTable, Status);

        /* Load OEM9 table (causes table override) */

        Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt3Code);
        AE_CHECK_OK (AcpiLoadTable, Status);
    }

    AeHardwareInterfaces ();
    AeGenericRegisters ();
    AeSetupConfiguration (Ssdt3Code);

    AeTestBufferArgument();
    AeTestPackageArgument ();
    AeMutexInterfaces ();

    /* Test _OSI install/remove */

    Status = AcpiInstallInterface ("");
    AE_CHECK_STATUS (AcpiInstallInterface, Status, AE_BAD_PARAMETER);

    Status = AcpiInstallInterface ("TestString");
    AE_CHECK_OK (AcpiInstallInterface, Status);

    Status = AcpiInstallInterface ("TestString");
    AE_CHECK_STATUS (AcpiInstallInterface, Status, AE_ALREADY_EXISTS);

    Status = AcpiRemoveInterface ("Windows 2006");
    AE_CHECK_OK (AcpiRemoveInterface, Status);

    Status = AcpiRemoveInterface ("TestString");
    AE_CHECK_OK (AcpiRemoveInterface, Status);

    Status = AcpiRemoveInterface ("XXXXXX");
    AE_CHECK_STATUS (AcpiRemoveInterface, Status, AE_NOT_EXIST);

    Status = AcpiInstallInterface ("AnotherTestString");
    AE_CHECK_OK (AcpiInstallInterface, Status);

    /* Test _OSI execution */

    Status = ExecuteOSI ("Extended Address Space Descriptor", 0xFFFFFFFF);
    AE_CHECK_OK (ExecuteOSI, Status);

    Status = ExecuteOSI ("Windows 2001", 0xFFFFFFFF);
    AE_CHECK_OK (ExecuteOSI, Status);

    Status = ExecuteOSI ("MichiganTerminalSystem", 0);
    AE_CHECK_OK (ExecuteOSI, Status);


    ReturnBuf.Length = 32;
    ReturnBuf.Pointer = Buffer;

    Status = AcpiGetName (ACPI_ROOT_OBJECT, ACPI_FULL_PATHNAME, &ReturnBuf);
    AE_CHECK_OK (AcpiGetName, Status);

    /* Get Devices */

    Status = AcpiGetDevices (NULL, AeGetDevices, NULL, NULL);
```

```
654     AE_CHECK_OK (AcpiGetDevices, Status);

656     Status = AcpiGetStatistics (&Stats);
657     AE_CHECK_OK (AcpiGetStatistics, Status);


660 #if (!ACPI_REDUCED_HARDWARE)

662     Status = AcpiInstallGlobalEventHandler (AeGlobalEventHandler, NULL);
663     AE_CHECK_OK (AcpiInstallGlobalEventHandler, Status);

665     /* If Hardware Reduced flag is set, we are all done */

667     if (AcpiGbl_ReducedHardware)
668     {
669         return;
670     }

672     Status = AcpiEnableEvent (ACPI_EVENT_GLOBAL, 0);
673     AE_CHECK_OK (AcpiEnableEvent, Status);

675     /*
676      * GPEs: Handlers, enable/disable, etc.
677      */
678     Status = AcpiInstallGpeHandler (NULL, 0, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
679     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

681     Status = AcpiEnableGpe (NULL, 0);
682     AE_CHECK_OK (AcpiEnableGpe, Status);

684     Status = AcpiRemoveGpeHandler (NULL, 0, AeGpeHandler);
685     AE_CHECK_OK (AcpiRemoveGpeHandler, Status);

687     Status = AcpiInstallGpeHandler (NULL, 0, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
688     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

690     Status = AcpiEnableGpe (NULL, 0);
691     AE_CHECK_OK (AcpiEnableGpe, Status);

693     Status = AcpiSetGpe (NULL, 0, ACPI_GPE_DISABLE);
694     AE_CHECK_OK (AcpiSetGpe, Status);

696     Status = AcpiSetGpe (NULL, 0, ACPI_GPE_ENABLE);
697     AE_CHECK_OK (AcpiSetGpe, Status);


700     Status = AcpiInstallGpeHandler (NULL, 1, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
701     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

703     Status = AcpiEnableGpe (NULL, 1);
704     AE_CHECK_OK (AcpiEnableGpe, Status);


707     Status = AcpiInstallGpeHandler (NULL, 2, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
708     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

710     Status = AcpiEnableGpe (NULL, 2);
711     AE_CHECK_OK (AcpiEnableGpe, Status);


714     Status = AcpiInstallGpeHandler (NULL, 3, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
715     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

717     Status = AcpiInstallGpeHandler (NULL, 4, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
718     AE_CHECK_OK (AcpiInstallGpeHandler, Status);
```



```
720     Status = AcpiInstallGpeHandler (NULL, 5, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
721     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

723     Status = AcpiGetHandle (NULL, "\\_SB", &Handle);
724     AE_CHECK_OK (AcpiGetHandle, Status);

726     Status = AcpiSetupGpeForWake (Handle, NULL, 5);
727     AE_CHECK_OK (AcpiSetupGpeForWake, Status);

729     Status = AcpiSetGpeWakeMask (NULL, 5, ACPI_GPE_ENABLE);
730     AE_CHECK_OK (AcpiSetGpeWakeMask, Status);

732     Status = AcpiSetupGpeForWake (Handle, NULL, 6);
733     AE_CHECK_OK (AcpiSetupGpeForWake, Status);

735     Status = AcpiSetupGpeForWake (ACPI_ROOT_OBJECT, NULL, 6);
736     AE_CHECK_OK (AcpiSetupGpeForWake, Status);

738     Status = AcpiSetupGpeForWake (Handle, NULL, 9);
739     AE_CHECK_OK (AcpiSetupGpeForWake, Status);

741     Status = AcpiInstallGpeHandler (NULL, 0x19, ACPI_GPE_LEVEL_TRIGGERED, AeGpeH
742     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

744     Status = AcpiEnableGpe (NULL, 0x19);
745     AE_CHECK_OK (AcpiEnableGpe, Status);


748     Status = AcpiInstallGpeHandler (NULL, 0x62, ACPI_GPE_LEVEL_TRIGGERED, AeGpeH
749     AE_CHECK_OK (AcpiInstallGpeHandler, Status);

751     Status = AcpiEnableGpe (NULL, 0x62);
752     AE_CHECK_OK (AcpiEnableGpe, Status);

754     Status = AcpiDisableGpe (NULL, 0x62);
755     AE_CHECK_OK (AcpiDisableGpe, Status);

757     AfInstallGpeBlock ();

759     /* Here is where the GPEs are actually "enabled" */

761     Status = AcpiUpdateAllGpes ();
762     AE_CHECK_OK (AcpiUpdateAllGpes, Status);

764     Status = AcpiGetHandle (NULL, "RSRC", &Handle);
765     if (ACPI_SUCCESS (Status))
766     {
767         ReturnBuf.Length = ACPI_ALLOCATE_BUFFER;

769         Status = AcpiGetVendorResource (Handle, "_CRS", &Uuid, &ReturnBuf);
770         if (ACPI_SUCCESS (Status))
771         {
772             AcpiOsFree (ReturnBuf.Pointer);
773         }
774     }

776     /* Test global lock */

778     Status = AcpiAcquireGlobalLock (0xFFFF, &LockHandle1);
779     AE_CHECK_OK (AcpiAcquireGlobalLock, Status);

781     Status = AcpiAcquireGlobalLock (0x5, &LockHandle2);
782     AE_CHECK_OK (AcpiAcquireGlobalLock, Status);

784     Status = AcpiReleaseGlobalLock (LockHandle1);
785     AE_CHECK_OK (AcpiReleaseGlobalLock, Status);
```

```
787     Status = AcpiReleaseGlobalLock (LockHandle2);
788     AE_CHECK_OK (AcpiReleaseGlobalLock, Status);

790 #endif /* !ACPI_REDUCED_HARDWARE */
791 }
```

```
**********************************************************
   44726 Thu Dec 26 13:50:08 2013
new/usr/src/common/acpica/tools/acpiexec/aehandlers.c
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aehandlers - Various handlers for acpiexec
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aecommon.h"

#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("aehandlers")

/* Local prototypes */

static void
AeNotifyHandler1 (
    ACPI_HANDLE             Device,
    UINT32                  Value,
    void                    *Context);

static void
```



```
AeNotifyHandler2 (
    ACPI_HANDLE             Device,
    UINT32                  Value,
    void                    *Context);

static void
AeCommonNotifyHandler (
    ACPI_HANDLE             Device,
    UINT32                  Value,
    UINT32                  HandlerId);

static void
AeDeviceNotifyHandler (
    ACPI_HANDLE             Device,
    UINT32                  Value,
    void                    *Context);

static ACPI_STATUS
AeExceptionHandler (
    ACPI_STATUS             AmlStatus,
    ACPI_NAME               Name,
    UINT16                  Opcode,
    UINT32                  AmlOffset,
    void                    *Context);

static ACPI_STATUS
AeTableHandler (
    UINT32                  Event,
    void                    *Table,
    void                    *Context);

static ACPI_STATUS
AeRegionInit (
    ACPI_HANDLE             RegionHandle,
    UINT32                  Function,
    void                    *HandlerContext,
    void                    **RegionContext);

static void
AeAttachedDataHandler (
    ACPI_HANDLE             Object,
    void                    *Data);

static void
AeAttachedDataHandler2 (
    ACPI_HANDLE             Object,
    void                    *Data);

static UINT32
AeInterfaceHandler (
    ACPI_STRING             InterfaceName,
    UINT32                  Supported);

static ACPI_STATUS
AeInstallEcHandler (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);

static ACPI_STATUS
AeInstallPciHandler (
    ACPI_HANDLE             ObjHandle,
    UINT32                  Level,
    void                    *Context,
    void                    **ReturnValue);
```

```
125 static ACPI_STATUS
126 AeInstallDeviceHandlers (
127     void);

129 #if (!ACPI_REDUCED_HARDWARE)
130 static UINT32
131 AeEventHandler (
132     void                    *Context);

134 static UINT32
135 AeSciHandler (
136     void                    *Context);

138 static char                 *TableEvents[] =
139 {
140     "LOAD",
141     "UNLOAD",
142     "UNKNOWN"
143 };
144 #endif /* !ACPI_REDUCED_HARDWARE */


147 static UINT32               SigintCount = 0;
148 static AE_DEBUG_REGIONS     AeRegions;
149 BOOLEAN                     AcpiGbl_DisplayRegionAccess = FALSE;

151 /*
152  * We will override some of the default region handlers, especially the
153  * SystemMemory handler, which must be implemented locally. Do not override
154  * the PCI_Config handler since we would like to exercise the default handler
155  * code. These handlers are installed "early" - before any _REG methods
156  * are executed - since they are special in the sense that the ACPI spec
157  * declares that they must "always be available". Cannot override the
158  * DataTable region handler either -- needed for test execution.
159  */
160 static ACPI_ADR_SPACE_TYPE  DefaultSpaceIdList[] =
161 {
162     ACPI_ADR_SPACE_SYSTEM_MEMORY,
163     ACPI_ADR_SPACE_SYSTEM_IO
164 };

166 /*
167  * We will install handlers for some of the various address space IDs.
168  * Test one user-defined address space (used by aslts).
169  */
170 #define ACPI_ADR_SPACE_USER_DEFINED1        0x80
171 #define ACPI_ADR_SPACE_USER_DEFINED2        0xE4

173 static ACPI_ADR_SPACE_TYPE  SpaceIdList[] =
174 {
175     ACPI_ADR_SPACE_SMBUS,
176     ACPI_ADR_SPACE_CMOS,
177     ACPI_ADR_SPACE_PCI_BAR_TARGET,
178     ACPI_ADR_SPACE_IPMI,
179     ACPI_ADR_SPACE_GPIO,
180     ACPI_ADR_SPACE_GSBUS,
181     ACPI_ADR_SPACE_FIXED_HARDWARE,
182     ACPI_ADR_SPACE_USER_DEFINED1,
183     ACPI_ADR_SPACE_USER_DEFINED2
184 };

186 static ACPI_CONNECTION_INFO   AeMyContext;


189 /******************************************************************************
```



```
190  *
191  * FUNCTION:    AeCtrlCHandler
192  *
193  * PARAMETERS:  Sig
194  *
195  * RETURN:      none
196  *
197  * DESCRIPTION: Control-C handler. Abort running control method if any.
198  *
199  *****************************************************************************/

201 void ACPI_SYSTEM_XFACE
202 AeCtrlCHandler (
203     int                     Sig)
204 {

206     signal (SIGINT, SIG_IGN);
207     SigintCount++;

209     AcpiOsPrintf ("Caught a ctrl-c (#%u)\n\n", SigintCount);

211     if (AcpiGbl_MethodExecuting)
212     {
213         AcpiGbl_AbortMethod = TRUE;
214         signal (SIGINT, AeCtrlCHandler);

216         if (SigintCount < 10)
217         {
218             return;
219         }
220     }

222     (void) AcpiOsTerminate ();
223     exit (0);
224 }


227 /******************************************************************************
228  *
229  * FUNCTION:    AeNotifyHandler(s)
230  *
231  * PARAMETERS:  Standard notify handler parameters
232  *
233  * RETURN:      Status
234  *
235  * DESCRIPTION: Notify handlers for AcpiExec utility. Used by the ASL
236  *              test suite(s) to communicate errors and other information to
237  *              this utility via the Notify() operator. Tests notify handling
238  *              and multiple notify handler support.
239  *
240  *****************************************************************************/

242 static void
243 AeNotifyHandler1 (
244     ACPI_HANDLE             Device,
245     UINT32                  Value,
246     void                    *Context)
247 {
248     AeCommonNotifyHandler (Device, Value, 1);
249 }

251 static void
252 AeNotifyHandler2 (
253     ACPI_HANDLE             Device,
254     UINT32                  Value,
255     void                    *Context)
```

```
256 {
257     AeCommonNotifyHandler (Device, Value, 2);
258 }

260 static void
261 AeCommonNotifyHandler (
262     ACPI_HANDLE             Device,
263     UINT32                  Value,
264     UINT32                  HandlerId)
265 {
266     char                    *Type;


269     Type = "Device";
270     if (Value <= ACPI_MAX_SYS_NOTIFY)
271     {
272         Type = "System";
273     }

275     switch (Value)
276     {
277 #if 0
278     case 0:

280         printf ("[AcpiExec] Method Error 0x%X: Results not equal\n", Value);
281         if (AcpiGbl_DebugFile)
282         {
283             AcpiOsPrintf ("[AcpiExec] Method Error: Results not equal\n");
284         }
285         break;

287     case 1:

289         printf ("[AcpiExec] Method Error: Incorrect numeric result\n");
290         if (AcpiGbl_DebugFile)
291         {
292             AcpiOsPrintf ("[AcpiExec] Method Error: Incorrect numeric result\n")
293         }
294         break;

296     case 2:

298         printf ("[AcpiExec] Method Error: An operand was overwritten\n");
299         if (AcpiGbl_DebugFile)
300         {
301             AcpiOsPrintf ("[AcpiExec] Method Error: An operand was overwritten\n
302         }
303         break;

305 #endif

307     default:

309         printf ("[AcpiExec] Handler %u: Received a %s Notify on [%4.4s] %p Value
310             HandlerId, Type, AcpiUtGetNodeName (Device), Device, Value,
311             AcpiUtGetNotifyName (Value));
312         if (AcpiGbl_DebugFile)
313         {
314             AcpiOsPrintf ("[AcpiExec] Handler %u: Received a %s notify, Value 0x
315                 HandlerId, Type, Value);
316         }

318         (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
319         break;
320     }
321 }
```



```


324 /*****************************************************************************
325  *
326  * FUNCTION:    AeSystemNotifyHandler
327  *
328  * PARAMETERS:  Standard notify handler parameters
329  *
330  * RETURN:      Status
331  *
332  * DESCRIPTION: System notify handler for AcpiExec utility. Used by the ASL
333  *              test suite(s) to communicate errors and other information to
334  *              this utility via the Notify() operator.
335  *
336  ****************************************************************************/

338 static void
339 AeSystemNotifyHandler (
340     ACPI_HANDLE                 Device,
341     UINT32                      Value,
342     void                        *Context)
343 {

345     printf ("[AcpiExec] Global:    Received a System Notify on [%4.4s] %p Value
346         AcpiUtGetNodeName (Device), Device, Value,
347         AcpiUtGetNotifyName (Value));
348     if (AcpiGbl_DebugFile)
349     {
350         AcpiOsPrintf ("[AcpiExec] Global:    Received a System Notify, Value 0x%
351     }

353     (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
354 }


357 /*****************************************************************************
358  *
359  * FUNCTION:    AeDeviceNotifyHandler
360  *
361  * PARAMETERS:  Standard notify handler parameters
362  *
363  * RETURN:      Status
364  *
365  * DESCRIPTION: Device notify handler for AcpiExec utility. Used by the ASL
366  *              test suite(s) to communicate errors and other information to
367  *              this utility via the Notify() operator.
368  *
369  ****************************************************************************/

371 static void
372 AeDeviceNotifyHandler (
373     ACPI_HANDLE                 Device,
374     UINT32                      Value,
375     void                        *Context)
376 {

378     printf ("[AcpiExec] Global:    Received a Device Notify on [%4.4s] %p Value
379         AcpiUtGetNodeName (Device), Device, Value,
380         AcpiUtGetNotifyName (Value));
381     if (AcpiGbl_DebugFile)
382     {
383         AcpiOsPrintf ("[AcpiExec] Global:    Received a Device Notify, Value 0x%
384     }

386     (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
387 }
```

```
390 /******************************************************************************
391  *
392  * FUNCTION:    AeExceptionHandler
393  *
394  * PARAMETERS:  Standard exception handler parameters
395  *
396  * RETURN:      Status
397  *
398  * DESCRIPTION: System exception handler for AcpiExec utility.
399  *
400  *****************************************************************************/

402 static ACPI_STATUS
403 AeExceptionHandler (
404     ACPI_STATUS             AmlStatus,
405     ACPI_NAME               Name,
406     UINT16                  Opcode,
407     UINT32                  AmlOffset,
408     void                    *Context)
409 {
410     ACPI_STATUS             NewAmlStatus = AmlStatus;
411     ACPI_STATUS             Status;
412     ACPI_BUFFER             ReturnObj;
413     ACPI_OBJECT_LIST        ArgList;
414     ACPI_OBJECT             Arg[3];
415     const char              *Exception;


418     Exception = AcpiFormatException (AmlStatus);
419     AcpiOsPrintf ("[AcpiExec] Exception %s during execution ", Exception);
420     if (Name)
421     {
422         AcpiOsPrintf ("of method [%4.4s]", (char *) &Name);
423     }
424     else
425     {
426         AcpiOsPrintf ("at module level (table load)");
427     }
428     AcpiOsPrintf (" Opcode [%s] @%X\n", AcpiPsGetOpcodeName (Opcode), AmlOffset)

430     /*
431      * Invoke the _ERR method if present
432      *
433      * Setup parameter object
434      */
435     ArgList.Count = 3;
436     ArgList.Pointer = Arg;

438     Arg[0].Type = ACPI_TYPE_INTEGER;
439     Arg[0].Integer.Value = AmlStatus;

441     Arg[1].Type = ACPI_TYPE_STRING;
442     Arg[1].String.Pointer = ACPI_CAST_PTR (char, Exception);
443     Arg[1].String.Length = ACPI_STRLEN (Exception);

445     Arg[2].Type = ACPI_TYPE_INTEGER;
446     Arg[2].Integer.Value = AcpiOsGetThreadId();

448     /* Setup return buffer */

450     ReturnObj.Pointer = NULL;
451     ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

453     Status = AcpiEvaluateObject (NULL, "\\_ERR", &ArgList, &ReturnObj);
```



```
454     if (ACPI_SUCCESS (Status))
455     {
456         if (ReturnObj.Pointer)
457         {
458             /* Override original status */

460             NewAmlStatus = (ACPI_STATUS)
461                 ((ACPI_OBJECT *) ReturnObj.Pointer)->Integer.Value;

463             /* Free a buffer created via ACPI_ALLOCATE_BUFFER */

465             AcpiOsFree (ReturnObj.Pointer);
466         }
467     }
468     else if (Status != AE_NOT_FOUND)
469     {
470         AcpiOsPrintf ("[AcpiExec] Could not execute _ERR method, %s\n",
471             AcpiFormatException (Status));
472     }

474     /* Global override */

476     if (AcpiGbl_IgnoreErrors)
477     {
478         NewAmlStatus = AE_OK;
479     }

481     if (NewAmlStatus != AmlStatus)
482     {
483         AcpiOsPrintf ("[AcpiExec] Exception override, new status %s\n",
484             AcpiFormatException (NewAmlStatus));
485     }

487     return (NewAmlStatus);
488 }


491 /******************************************************************************
492  *
493  * FUNCTION:    AeTableHandler
494  *
495  * PARAMETERS:  Table handler
496  *
497  * RETURN:      Status
498  *
499  * DESCRIPTION: System table handler for AcpiExec utility.
500  *
501  *****************************************************************************/

503 static ACPI_STATUS
504 AeTableHandler (
505     UINT32                  Event,
506     void                    *Table,
507     void                    *Context)
508 {
509 #if (!ACPI_REDUCED_HARDWARE)
510     ACPI_STATUS             Status;
511 #endif /* !ACPI_REDUCED_HARDWARE */


514     if (Event > ACPI_NUM_TABLE_EVENTS)
515     {
516         Event = ACPI_NUM_TABLE_EVENTS;
517     }

519 #if (!ACPI_REDUCED_HARDWARE)
```

```
520     /* Enable any GPEs associated with newly-loaded GPE methods */

522     Status = AcpiUpdateAllGpes ();
523     AE_CHECK_OK (AcpiUpdateAllGpes, Status);

525     printf ("[AcpiExec] Table Event %s, [%4.4s] %p\n",
526         TableEvents[Event], ((ACPI_TABLE_HEADER *) Table)->Signature, Table);
527 #endif /* !ACPI_REDUCED_HARDWARE */

529     return (AE_OK);
530 }


533 /******************************************************************************
534  *
535  * FUNCTION:    AeGpeHandler
536  *
537  * DESCRIPTION: Common GPE handler for acpiexec
538  *
539  *****************************************************************************/

541 UINT32
542 AeGpeHandler (
543     ACPI_HANDLE             GpeDevice,
544     UINT32                  GpeNumber,
545     void                    *Context)
546 {
547     ACPI_NAMESPACE_NODE     *DeviceNode = (ACPI_NAMESPACE_NODE *) GpeDevice;


550     AcpiOsPrintf ("[AcpiExec] GPE Handler received GPE%02X (GPE block %4.4s)\n",
551         GpeNumber, GpeDevice ? DeviceNode->Name.Ascii : "FADT");

553     return (ACPI_REENABLE_GPE);
554 }


557 /******************************************************************************
558  *
559  * FUNCTION:    AeGlobalEventHandler
560  *
561  * DESCRIPTION: Global GPE/Fixed event handler
562  *
563  *****************************************************************************/

565 void
566 AeGlobalEventHandler (
567     UINT32                  Type,
568     ACPI_HANDLE             Device,
569     UINT32                  EventNumber,
570     void                    *Context)
571 {
572     char                    *TypeName;


575     switch (Type)
576     {
577     case ACPI_EVENT_TYPE_GPE:

579         TypeName = "GPE";
580         break;

582     case ACPI_EVENT_TYPE_FIXED:

584         TypeName = "FixedEvent";
585         break;
```



```
587     default:

589         TypeName = "UNKNOWN";
590         break;
591     }

593     AcpiOsPrintf ("[AcpiExec] Global Event Handler received: Type %s Number %.2X
594         TypeName, EventNumber, Device);
595 }


598 /******************************************************************************
599  *
600  * FUNCTION:    AeAttachedDataHandler
601  *
602  * DESCRIPTION: Handler for deletion of nodes with attached data (attached via
603  *              AcpiAttachData)
604  *
605  *****************************************************************************/

607 static void
608 AeAttachedDataHandler (
609     ACPI_HANDLE             Object,
610     void                    *Data)
611 {
612     ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Data);


615     AcpiOsPrintf ("Received an attached data deletion (1) on %4.4s\n",
616         Node->Name.Ascii);
617 }


620 /******************************************************************************
621  *
622  * FUNCTION:    AeAttachedDataHandler2
623  *
624  * DESCRIPTION: Handler for deletion of nodes with attached data (attached via
625  *              AcpiAttachData)
626  *
627  *****************************************************************************/

629 static void
630 AeAttachedDataHandler2 (
631     ACPI_HANDLE             Object,
632     void                    *Data)
633 {
634     ACPI_NAMESPACE_NODE     *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Data);


637     AcpiOsPrintf ("Received an attached data deletion (2) on %4.4s\n",
638         Node->Name.Ascii);
639 }


642 /******************************************************************************
643  *
644  * FUNCTION:    AeInterfaceHandler
645  *
646  * DESCRIPTION: Handler for _OSI invocations
647  *
648  *****************************************************************************/

650 static UINT32
651 AeInterfaceHandler (
```

```
652     ACPI_STRING             InterfaceName,
653     UINT32                  Supported)
654 {
655     ACPI_FUNCTION_NAME (AeInterfaceHandler);


658     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
659         "Received _OSI (\"%s\"), is %ssupported\n",
660         InterfaceName, Supported == 0 ? "not " : ""));

662     return (Supported);
663 }


666 #if (!ACPI_REDUCED_HARDWARE)
667 /******************************************************************************
668  *
669  * FUNCTION:    AeEventHandler, AeSciHandler
670  *
671  * DESCRIPTION: Handler for Fixed Events and SCIs
672  *
673  *****************************************************************************/

675 static UINT32
676 AeEventHandler (
677     void                    *Context)
678 {
679     return (0);
680 }

682 static UINT32
683 AeSciHandler (
684     void                    *Context)
685 {

687     AcpiOsPrintf ("[AcpiExec] Received an SCI at handler\n");
688     return (0);
689 }

691 #endif /* !ACPI_REDUCED_HARDWARE */


694 /******************************************************************************
695  *
696  * FUNCTION:    AeRegionInit
697  *
698  * PARAMETERS:  None
699  *
700  * RETURN:      Status
701  *
702  * DESCRIPTION: Opregion init function.
703  *
704  *****************************************************************************/

706 static ACPI_STATUS
707 AeRegionInit (
708     ACPI_HANDLE             RegionHandle,
709     UINT32                  Function,
710     void                    *HandlerContext,
711     void                    **RegionContext)
712 {

714     if (Function == ACPI_REGION_DEACTIVATE)
715     {
716         *RegionContext = NULL;
717     }
```



```
718     else
719     {
720         *RegionContext = RegionHandle;
721     }

723     return (AE_OK);
724 }


727 /******************************************************************************
728  *
729  * FUNCTION:    AeInstallSciHandler
730  *
731  * PARAMETERS:  None
732  *
733  * RETURN:      Status
734  *
735  * DESCRIPTION: Install handler for SCIs. Exercise the code by doing an
736  *              install/remove/install.
737  *
738  *****************************************************************************/

740 static ACPI_STATUS
741 AeInstallSciHandler (
742     void)
743 {
744     ACPI_STATUS             Status;


747     Status = AcpiInstallSciHandler (AeSciHandler, &AeMyContext);
748     if (ACPI_FAILURE (Status))
749     {
750         ACPI_EXCEPTION ((AE_INFO, Status,
751             "Could not install an SCI handler (1)"));
752     }

754     Status = AcpiRemoveSciHandler (AeSciHandler);
755     if (ACPI_FAILURE (Status))
756     {
757         ACPI_EXCEPTION ((AE_INFO, Status,
758             "Could not remove an SCI handler"));
759     }

761     Status = AcpiInstallSciHandler (AeSciHandler, &AeMyContext);
762     if (ACPI_FAILURE (Status))
763     {
764         ACPI_EXCEPTION ((AE_INFO, Status,
765             "Could not install an SCI handler (2)"));
766     }

768     return (Status);
769 }


772 /******************************************************************************
773  *
774  * FUNCTION:    AeInstallDeviceHandlers, AeInstallEcHandler,
775  *              AeInstallPciHandler
776  *
777  * PARAMETERS:  ACPI_WALK_NAMESPACE callback
778  *
779  * RETURN:      Status
780  *
781  * DESCRIPTION: Walk entire namespace, install a handler for every EC
782  *              and PCI device found.
783  *
```

```
784  *****************************************************************************/

786 static ACPI_STATUS
787 AeInstallEcHandler (
788     ACPI_HANDLE             ObjHandle,
789     UINT32                  Level,
790     void                    *Context,
791     void                    **ReturnValue)
792 {
793     ACPI_STATUS             Status;


796     /* Install the handler for this EC device */

798     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_EC,
799         AeRegionHandler, AeRegionInit, &AeMyContext);
800     if (ACPI_FAILURE (Status))
801     {
802         ACPI_EXCEPTION ((AE_INFO, Status,
803             "Could not install an OpRegion handler for EC device (%p)",
804             ObjHandle));
805     }

807     return (Status);
808 }

810 static ACPI_STATUS
811 AeInstallPciHandler (
812     ACPI_HANDLE             ObjHandle,
813     UINT32                  Level,
814     void                    *Context,
815     void                    **ReturnValue)
816 {
817     ACPI_STATUS             Status;


820     /* Install memory and I/O handlers for the PCI device */

822     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_SYSTEM_IO
823         AeRegionHandler, AeRegionInit, &AeMyContext);
824     if (ACPI_FAILURE (Status))
825     {
826         ACPI_EXCEPTION ((AE_INFO, Status,
827             "Could not install an OpRegion handler for PCI device (%p)",
828             ObjHandle));
829     }

831     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_SYSTEM_ME
832         AeRegionHandler, AeRegionInit, &AeMyContext);
833     if (ACPI_FAILURE (Status))
834     {
835         ACPI_EXCEPTION ((AE_INFO, Status,
836             "Could not install an OpRegion handler for PCI device (%p)",
837             ObjHandle));
838     }

840     return (AE_CTRL_TERMINATE);
841 }

843 static ACPI_STATUS
844 AeInstallDeviceHandlers (
845     void)
846 {

848     /* Find all Embedded Controller devices */
```



```
850     AcpiGetDevices ("PNP0C09", AeInstallEcHandler, NULL, NULL);

852     /* Install a PCI handler */

854     AcpiGetDevices ("PNP0A08", AeInstallPciHandler, NULL, NULL);
855     return (AE_OK);
856 }


859 /*****************************************************************************
860  *
861  * FUNCTION:    AeInstallLateHandlers
862  *
863  * PARAMETERS:  None
864  *
865  * RETURN:      Status
866  *
867  * DESCRIPTION: Install handlers for the AcpiExec utility.
868  *
869  ****************************************************************************/

871 ACPI_STATUS
872 AeInstallLateHandlers (
873     void)
874 {
875     ACPI_STATUS             Status;
876     UINT32                  i;


879 #if (!ACPI_REDUCED_HARDWARE)
880     if (!AcpiGbl_ReducedHardware)
881     {
882         /* Install a user SCI handler */

884         Status = AeInstallSciHandler ();
885         AE_CHECK_OK (AeInstallSciHandler, Status);

887         /* Install some fixed event handlers */

889         Status = AcpiInstallFixedEventHandler (ACPI_EVENT_GLOBAL, AeEventHandler
890         AE_CHECK_OK (AcpiInstallFixedEventHandler, Status);

892         Status = AcpiInstallFixedEventHandler (ACPI_EVENT_RTC, AeEventHandler, N
893         AE_CHECK_OK (AcpiInstallFixedEventHandler, Status);
894     }
895 #endif /* !ACPI_REDUCED_HARDWARE */

897     AeMyContext.Connection = NULL;
898     AeMyContext.AccessLength = 0xA5;

900     /*
901      * We will install a handler for each EC device, directly under the EC
902      * device definition. This is unlike the other handlers which we install
903      * at the root node. Also install memory and I/O handlers at any PCI
904      * devices.
905      */
906     AeInstallDeviceHandlers ();

908     /*
909      * Install handlers for some of the "device driver" address spaces
910      * such as SMBus, etc.
911      */
912     for (i = 0; i < ACPI_ARRAY_LENGTH (SpaceIdList); i++)
913     {
914         /* Install handler at the root object */
```

```
916         Status = AcpiInstallAddressSpaceHandler (ACPI_ROOT_OBJECT,
917                     SpaceIdList[i], AeRegionHandler,
918                     AeRegionInit, &AeMyContext);
919         if (ACPI_FAILURE (Status))
920         {
921             ACPI_EXCEPTION ((AE_INFO, Status,
922                 "Could not install an OpRegion handler for %s space(%u)",
923                 AcpiUtGetRegionName((UINT8) SpaceIdList[i]), SpaceIdList[i]));
924             return (Status);
925         }
926     }

928     return (AE_OK);
929 }


932 /******************************************************************************
933  *
934  * FUNCTION:    AeInstallEarlyHandlers
935  *
936  * PARAMETERS:  None
937  *
938  * RETURN:      Status
939  *
940  * DESCRIPTION: Install handlers for the AcpiExec utility.
941  *
942  * Notes:       Don't install handler for PCI_Config, we want to use the
943  *              default handler to exercise that code.
944  *
945  *****************************************************************************/

947 ACPI_STATUS
948 AeInstallEarlyHandlers (
949     void)
950 {
951     ACPI_STATUS             Status;
952     UINT32                  i;
953     ACPI_HANDLE             Handle;


956     ACPI_FUNCTION_ENTRY ();


959     Status = AcpiInstallInterfaceHandler (AeInterfaceHandler);
960     if (ACPI_FAILURE (Status))
961     {
962         printf ("Could not install interface handler, %s\n",
963             AcpiFormatException (Status));
964     }

966     Status = AcpiInstallTableHandler (AeTableHandler, NULL);
967     if (ACPI_FAILURE (Status))
968     {
969         printf ("Could not install table handler, %s\n",
970             AcpiFormatException (Status));
971     }

973     Status = AcpiInstallExceptionHandler (AeExceptionHandler);
974     if (ACPI_FAILURE (Status))
975     {
976         printf ("Could not install exception handler, %s\n",
977             AcpiFormatException (Status));
978     }

980     /* Install global notify handlers */
```



```
982     Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_SYSTEM_NOTIFY,
983         AeSystemNotifyHandler, NULL);
984     if (ACPI_FAILURE (Status))
985     {
986         printf ("Could not install a global system notify handler, %s\n",
987             AcpiFormatException (Status));
988     }

990     Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_DEVICE_NOTIFY,
991         AeDeviceNotifyHandler, NULL);
992     if (ACPI_FAILURE (Status))
993     {
994         printf ("Could not install a global notify handler, %s\n",
995             AcpiFormatException (Status));
996     }

998     Status = AcpiGetHandle (NULL, "\\_SB", &Handle);
999     if (ACPI_SUCCESS (Status))
1000    {
1001        Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1002            AeNotifyHandler1, NULL);
1003        if (ACPI_FAILURE (Status))
1004        {
1005            printf ("Could not install a notify handler, %s\n",
1006                AcpiFormatException (Status));
1007        }

1009        Status = AcpiRemoveNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1010            AeNotifyHandler1);
1011        if (ACPI_FAILURE (Status))
1012        {
1013            printf ("Could not remove a notify handler, %s\n",
1014                AcpiFormatException (Status));
1015        }

1017        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1018            AeNotifyHandler1, NULL);
1019        AE_CHECK_OK (AcpiInstallNotifyHandler, Status);

1021        Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1022            AeNotifyHandler1);
1023        AE_CHECK_OK (AcpiRemoveNotifyHandler, Status);

1025 #if 0
1026        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1027            AeNotifyHandler1, NULL);
1028        if (ACPI_FAILURE (Status))
1029        {
1030            printf ("Could not install a notify handler, %s\n",
1031                AcpiFormatException (Status));
1032        }
1033 #endif

1035        /* Install two handlers for _SB_ */

1037        Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1038            AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));

1040        Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1041            AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));

1043        /* Attempt duplicate handler installation, should fail */

1045        Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1046            AeNotifyHandler1, ACPI_CAST_PTR (void, 0x77777777));
```

```
        Status = AcpiAttachData (Handle, AeAttachedDataHandler, Handle);
        AE_CHECK_OK (AcpiAttachData, Status);

        Status = AcpiDetachData (Handle, AeAttachedDataHandler);
        AE_CHECK_OK (AcpiDetachData, Status);

        Status = AcpiAttachData (Handle, AeAttachedDataHandler, Handle);
        AE_CHECK_OK (AcpiAttachData, Status);

        /* Test support for multiple attaches */

        Status = AcpiAttachData (Handle, AeAttachedDataHandler2, Handle);
        AE_CHECK_OK (AcpiAttachData, Status);
    }
    else
    {
        printf ("No _SB_ found, %s\n", AcpiFormatException (Status));
    }


    Status = AcpiGetHandle (NULL, "\\_TZ.TZ1", &Handle);
    if (ACPI_SUCCESS (Status))
    {
        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));

        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));

        Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler1);
        Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler2);

        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));

        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));
    }

    Status = AcpiGetHandle (NULL, "\\_PR.CPU0", &Handle);
    if (ACPI_SUCCESS (Status))
    {
        Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
            AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));

        Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
            AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));
    }

    /*
     * Install handlers that will override the default handlers for some of
     * the space IDs.
     */
    for (i = 0; i < ACPI_ARRAY_LENGTH (DefaultSpaceIdList); i++)
    {
        /* Install handler at the root object */

        Status = AcpiInstallAddressSpaceHandler (ACPI_ROOT_OBJECT,
                    DefaultSpaceIdList[i], AeRegionHandler,
                    AeRegionInit, &AeMyContext);
        if (ACPI_FAILURE (Status))
        {
            ACPI_EXCEPTION ((AE_INFO, Status,
                "Could not install a default OpRegion handler for %s space(%u)",
```



```
                    AcpiUtGetRegionName ((UINT8) DefaultSpaceIdList[i]),
                    DefaultSpaceIdList[i]));
                return (Status);
            }
    }

    /*
     * Initialize the global Region Handler space
     * MCW 3/23/00
     */
    AeRegions.NumberOfRegions = 0;
    AeRegions.RegionList = NULL;
    return (Status);
}


/*****************************************************************************
 *
 * FUNCTION:    AeRegionHandler
 *
 * PARAMETERS:  Standard region handler parameters
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Test handler - Handles some dummy regions via memory that can
 *              be manipulated in Ring 3. Simulates actual reads and writes.
 *
 ****************************************************************************/

ACPI_STATUS
AeRegionHandler (
    UINT32                  Function,
    ACPI_PHYSICAL_ADDRESS   Address,
    UINT32                  BitWidth,
    UINT64                  *Value,
    void                    *HandlerContext,
    void                    *RegionContext)
{

    ACPI_OPERAND_OBJECT     *RegionObject = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT,
    UINT8                   *Buffer = ACPI_CAST_PTR (UINT8, Value);
    ACPI_PHYSICAL_ADDRESS   BaseAddress;
    ACPI_SIZE               Length;
    BOOLEAN                 BufferExists;
    AE_REGION               *RegionElement;
    void                    *BufferValue;
    ACPI_STATUS             Status;
    UINT32                  ByteWidth;
    UINT32                  i;
    UINT8                   SpaceId;
    ACPI_CONNECTION_INFO    *MyContext;
    UINT32                  Value1;
    UINT32                  Value2;
    ACPI_RESOURCE           *Resource;


    ACPI_FUNCTION_NAME (AeRegionHandler);

    /*
     * If the object is not a region, simply return
     */
    if (RegionObject->Region.Type != ACPI_TYPE_REGION)
    {
        return (AE_OK);
    }
```

```
1180     /* Check that we actually got back our context parameter */

1182     if (HandlerContext != &AeMyContext)
1183     {
1184         printf ("Region handler received incorrect context %p, should be %p\n",
1185             HandlerContext, &AeMyContext);
1186     }

1188     MyContext = ACPI_CAST_PTR (ACPI_CONNECTION_INFO, HandlerContext);

1190     /*
1191      * Find the region's address space and length before searching
1192      * the linked list.
1193      */
1194     BaseAddress = RegionObject->Region.Address;
1195     Length = (ACPI_SIZE) RegionObject->Region.Length;
1196     SpaceId = RegionObject->Region.SpaceId;

1198     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION, "Operation Region request on %s at 0x%X
1199             AcpiUtGetRegionName (RegionObject->Region.SpaceId),
1200             (UINT32) Address));

1202     /*
1203      * Region support can be disabled with the -do option.
1204      * We use this to support dynamically loaded tables where we pass a valid
1205      * address to the AML.
1206      */
1207     if (AcpiGbl_DbOpt_NoRegionSupport)
1208     {
1209         BufferValue = ACPI_TO_POINTER (Address);
1210         ByteWidth = (BitWidth / 8);

1212         if (BitWidth % 8)
1213         {
1214             ByteWidth += 1;
1215         }
1216         goto DoFunction;
1217     }

1219     switch (SpaceId)
1220     {
1221     case ACPI_ADR_SPACE_SYSTEM_IO:
1222         /*
1223          * For I/O space, exercise the port validation
1224          * Note: ReadPort currently always returns all ones, length=BitLength
1225          */
1226         switch (Function & ACPI_IO_MASK)
1227         {
1228         case ACPI_READ:

1230             if (BitWidth == 64)
1231             {
1232                 /* Split the 64-bit request into two 32-bit requests */

1234                 Status = AcpiHwReadPort (Address, &Value1, 32);
1235                 AE_CHECK_OK (AcpiHwReadPort, Status);
1236                 Status = AcpiHwReadPort (Address+4, &Value2, 32);
1237                 AE_CHECK_OK (AcpiHwReadPort, Status);

1239                 *Value = Value1 | ((UINT64) Value2 << 32);
1240             }
1241             else
1242             {
1243                 Status = AcpiHwReadPort (Address, &Value1, BitWidth);
1244                 AE_CHECK_OK (AcpiHwReadPort, Status);
1245                 *Value = (UINT64) Value1;
```



```
1246             }
1247             break;

1249         case ACPI_WRITE:

1251             if (BitWidth == 64)
1252             {
1253                 /* Split the 64-bit request into two 32-bit requests */

1255                 Status = AcpiHwWritePort (Address, ACPI_LODWORD (*Value), 32);
1256                 AE_CHECK_OK (AcpiHwWritePort, Status);
1257                 Status = AcpiHwWritePort (Address+4, ACPI_HIDWORD (*Value), 32);
1258                 AE_CHECK_OK (AcpiHwWritePort, Status);
1259             }
1260             else
1261             {
1262                 Status = AcpiHwWritePort (Address, (UINT32) *Value, BitWidth);
1263                 AE_CHECK_OK (AcpiHwWritePort, Status);
1264             }
1265             break;

1267         default:

1269             Status = AE_BAD_PARAMETER;
1270             break;
1271         }

1273         if (ACPI_FAILURE (Status))
1274         {
1275             return (Status);
1276         }

1278         /* Now go ahead and simulate the hardware */
1279         break;

1281     /*
1282      * SMBus and GenericSerialBus support the various bidirectional
1283      * protocols.
1284      */
1285     case ACPI_ADR_SPACE_SMBUS:
1286     case ACPI_ADR_SPACE_GSBUS:  /* ACPI 5.0 */

1288         Length = 0;

1290         switch (Function & ACPI_IO_MASK)
1291         {
1292         case ACPI_READ:

1294             switch (Function >> 16)
1295             {
1296             case AML_FIELD_ATTRIB_QUICK:
1297             case AML_FIELD_ATTRIB_SEND_RCV:
1298             case AML_FIELD_ATTRIB_BYTE:

1300                 Length = 1;
1301                 break;

1303             case AML_FIELD_ATTRIB_WORD:
1304             case AML_FIELD_ATTRIB_WORD_CALL:

1306                 Length = 2;
1307                 break;

1309             case AML_FIELD_ATTRIB_BLOCK:
1310             case AML_FIELD_ATTRIB_BLOCK_CALL:
```

```
1312                Length = 32;
1313                break;

1315            case AML_FIELD_ATTRIB_MULTIBYTE:
1316            case AML_FIELD_ATTRIB_RAW_BYTES:
1317            case AML_FIELD_ATTRIB_RAW_PROCESS:

1319                /* (-2) for status/length */
1320                Length = MyContext->AccessLength - 2;
1321                break;

1323            default:

1325                break;
1326            }
1327            break;

1329        case ACPI_WRITE:

1331            switch (Function >> 16)
1332            {
1333            case AML_FIELD_ATTRIB_QUICK:
1334            case AML_FIELD_ATTRIB_SEND_RCV:
1335            case AML_FIELD_ATTRIB_BYTE:
1336            case AML_FIELD_ATTRIB_WORD:
1337            case AML_FIELD_ATTRIB_BLOCK:

1339                Length = 0;
1340                break;

1342            case AML_FIELD_ATTRIB_WORD_CALL:
1343                Length = 2;
1344                break;

1346            case AML_FIELD_ATTRIB_BLOCK_CALL:
1347                Length = 32;
1348                break;

1350            case AML_FIELD_ATTRIB_MULTIBYTE:
1351            case AML_FIELD_ATTRIB_RAW_BYTES:
1352            case AML_FIELD_ATTRIB_RAW_PROCESS:

1354                /* (-2) for status/length */
1355                Length = MyContext->AccessLength - 2;
1356                break;

1358            default:

1360                break;
1361            }
1362            break;

1364        default:

1366            break;
1367        }

1369        if (AcpiGbl_DisplayRegionAccess)
1370        {
1371            AcpiOsPrintf ("AcpiExec: %s "
1372                "%s: Attr %X Addr %.4X BaseAddr %.4X Len %.2X Width %X BufLen %X
1373                AcpiUtGetRegionName (SpaceId),
1374                (Function & ACPI_IO_MASK) ? "Write" : "Read ",
1375                (UINT32) (Function >> 16),
1376                (UINT32) Address, (UINT32) BaseAddress,
1377                Length, BitWidth, Buffer[1]);
```



```
1379            /* GenericSerialBus has a Connection() parameter */

1381            if (SpaceId == ACPI_ADR_SPACE_GSBUS)
1382            {
1383                Status = AcpiBufferToResource (MyContext->Connection,
1384                    MyContext->Length, &Resource);

1386                AcpiOsPrintf (" [AccLen %.2X Conn %p]",
1387                    MyContext->AccessLength, MyContext->Connection);
1388            }
1389            AcpiOsPrintf ("\n");
1390        }

1392        /* Setup the return buffer. Note: ASLTS depends on these fill values */

1394        for (i = 0; i < Length; i++)
1395        {
1396            Buffer[i+2] = (UINT8) (0xA0 + i);
1397        }

1399        Buffer[0] = 0x7A;
1400        Buffer[1] = (UINT8) Length;
1401        return (AE_OK);


1404    case ACPI_ADR_SPACE_IPMI: /* ACPI 4.0 */

1406        if (AcpiGbl_DisplayRegionAccess)
1407        {
1408            AcpiOsPrintf ("AcpiExec: IPMI "
1409                "%s: Attr %X Addr %.4X BaseAddr %.4X Len %.2X Width %X BufLen %X
1410                (Function & ACPI_IO_MASK) ? "Write" : "Read ",
1411                (UINT32) (Function >> 16), (UINT32) Address, (UINT32) BaseAddres
1412                Length, BitWidth, Buffer[1]);
1413        }

1415        /*
1416         * Regardless of a READ or WRITE, this handler is passed a 66-byte
1417         * buffer in which to return the IPMI status/length/data.
1418         *
1419         * Return some example data to show use of the bidirectional buffer
1420         */
1421        Buffer[0] = 0;       /* Status byte */
1422        Buffer[1] = 64;      /* Return buffer data length */
1423        Buffer[2] = 0;       /* Completion code */
1424        Buffer[3] = 0;       /* Reserved */

1426        /*
1427         * Fill the 66-byte buffer with the return data.
1428         * Note: ASLTS depends on these fill values.
1429         */
1430        for (i = 4; i < 66; i++)
1431        {
1432            Buffer[i] = (UINT8) (i);
1433        }
1434        return (AE_OK);

1436    default:
1437        break;
1438    }

1440    /*
1441     * Search through the linked list for this region's buffer
1442     */
1443    BufferExists = FALSE;
```

```
    RegionElement = AeRegions.RegionList;

    if (AeRegions.NumberOfRegions)
    {
        while (!BufferExists && RegionElement)
        {
            if (RegionElement->Address == BaseAddress &&
                RegionElement->Length == Length &&
                RegionElement->SpaceId == SpaceId)
            {
                BufferExists = TRUE;
            }
            else
            {
                RegionElement = RegionElement->NextRegion;
            }
        }
    }

    /*
     * If the Region buffer does not exist, create it now
     */
    if (!BufferExists)
    {
        /*
         * Do the memory allocations first
         */
        RegionElement = AcpiOsAllocate (sizeof (AE_REGION));
        if (!RegionElement)
        {
            return (AE_NO_MEMORY);
        }

        RegionElement->Buffer = AcpiOsAllocate (Length);
        if (!RegionElement->Buffer)
        {
            AcpiOsFree (RegionElement);
            return (AE_NO_MEMORY);
        }

        /* Initialize the region with the default fill value */

        ACPI_MEMSET (RegionElement->Buffer, AcpiGbl_RegionFillValue, Length);

        RegionElement->Address      = BaseAddress;
        RegionElement->Length       = Length;
        RegionElement->SpaceId      = SpaceId;
        RegionElement->NextRegion   = NULL;

        /*
         * Increment the number of regions and put this one
         *  at the head of the list as it will probably get accessed
         *  more often anyway.
         */
        AeRegions.NumberOfRegions += 1;

        if (AeRegions.RegionList)
        {
            RegionElement->NextRegion = AeRegions.RegionList;
        }

        AeRegions.RegionList = RegionElement;
    }

    /*
     * Calculate the size of the memory copy
```



```
     */
    ByteWidth = (BitWidth / 8);

    if (BitWidth % 8)
    {
        ByteWidth += 1;
    }

    /*
     * The buffer exists and is pointed to by RegionElement.
     * We now need to verify the request is valid and perform the operation.
     *
     * NOTE: RegionElement->Length is in bytes, therefore it we compare against
     * ByteWidth (see above)
     */
    if (((UINT64) Address + ByteWidth) >
        ((UINT64)(RegionElement->Address) + RegionElement->Length))
    {
        ACPI_WARNING ((AE_INFO,
            "Request on [%4.4s] is beyond region limit Req-0x%X+0x%X, Base=0x%X,
            (RegionObject->Region.Node)->Name.Ascii, (UINT32) Address,
            ByteWidth, (UINT32)(RegionElement->Address),
            RegionElement->Length));

        return (AE_AML_REGION_LIMIT);
    }

    /*
     * Get BufferValue to point to the "address" in the buffer
     */
    BufferValue = ((UINT8 *) RegionElement->Buffer +
                      ((UINT64) Address - (UINT64) RegionElement->Address));

DoFunction:
    /*
     * Perform a read or write to the buffer space
     */
    switch (Function)
    {
    case ACPI_READ:
        /*
         * Set the pointer Value to whatever is in the buffer
         */
        ACPI_MEMCPY (Value, BufferValue, ByteWidth);
        break;

    case ACPI_WRITE:
        /*
         * Write the contents of Value to the buffer
         */
        ACPI_MEMCPY (BufferValue, Value, ByteWidth);
        break;

    default:

        return (AE_BAD_PARAMETER);
    }

    if (AcpiGbl_DisplayRegionAccess)
    {
        switch (SpaceId)
        {
        case ACPI_ADR_SPACE_SYSTEM_MEMORY:

            AcpiOsPrintf ("AcpiExec: SystemMemory "
                "%s: Val %.8X Addr %.4X Width %X [REGION: BaseAddr %.4X Len %.2X
```

```
1576                 (Function & ACPI_IO_MASK) ? "Write" : "Read ",
1577                 (UINT32) *Value, (UINT32) Address, BitWidth, (UINT32) BaseAddres
1578             break;

1580         case ACPI_ADR_SPACE_GPIO:   /* ACPI 5.0 */

1582             /* This space is required to always be ByteAcc */

1584             Status = AcpiBufferToResource (MyContext->Connection,
1585                 MyContext->Length, &Resource);

1587             AcpiOsPrintf ("AcpiExec: GeneralPurposeIo "
1588                 "%s: Val %.8X Addr %.4X BaseAddr %.4X Len %.2X Width %X AccLen %
1589                 (Function & ACPI_IO_MASK) ? "Write" : "Read ", (UINT32) *Value,
1590                 (UINT32) Address, (UINT32) BaseAddress, Length, BitWidth,
1591                 MyContext->AccessLength, MyContext->Connection);
1592             break;

1594         default:

1596             break;
1597         }
1598     }

1600     return (AE_OK);
1601 }
```

```
**********************************************************
   16170 Thu Dec 26 13:50:09 2013
new/usr/src/common/acpica/tools/acpiexec/aemain.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: aemain - Main routine for the AcpiExec utility
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #include "aecommon.h"

  46 #define _COMPONENT          ACPI_TOOLS
  47         ACPI_MODULE_NAME    ("aemain")


  50 /*
  51  * Main routine for the ACPI user-space execution utility.
  52  *
  53  * Portability note: The utility depends upon the host for command-line
  54  * wildcard support - it is not implemented locally. For example:
  55  *
  56  * Linux/Unix systems: Shell expands wildcards automatically.
  57  *
  58  * Windows: The setargv.obj module must be linked in to automatically
  59  * expand wildcards.
```



```
  60  */

  62 extern BOOLEAN              AcpiGbl_DebugTimeout;

  64 /* Local prototypes */

  66 static int
  67 AeDoOptions (
  68     int                     argc,
  69     char                    **argv);

  71 static ACPI_STATUS
  72 AcpiDbRunBatchMode (
  73     void);


  76 #define AE_BUFFER_SIZE              1024
  77 #define ASL_MAX_FILES               256

  79 /* Execution modes */

  81 #define AE_MODE_COMMAND_LOOP        0   /* Normal command execution loop */
  82 #define AE_MODE_BATCH_MULTIPLE      1   /* -b option to execute a command line *
  83 #define AE_MODE_BATCH_SINGLE        2   /* -m option to execute a single control


  86 /* Globals */

  88 UINT8                       AcpiGbl_RegionFillValue = 0;
  89 BOOLEAN                     AcpiGbl_IgnoreErrors = FALSE;
  90 BOOLEAN                     AcpiGbl_DbOpt_NoRegionSupport = FALSE;
  91 UINT8                       AcpiGbl_UseHwReducedFadt = FALSE;
  92 BOOLEAN                     AcpiGbl_DoInterfaceTests = FALSE;
  93 static UINT8                AcpiGbl_ExecutionMode = AE_MODE_COMMAND_LOOP;
  94 static char                 BatchBuffer[AE_BUFFER_SIZE];    /* Batch command buf
  95 static AE_TABLE_DESC        *AeTableListHead = NULL;

  97 #define ACPIEXEC_NAME               "AML Execution/Debug Utility"
  98 #define AE_SUPPORTED_OPTIONS        "?b:d:e:f:ghm^orv^:x:"


 101 /******************************************************************************
 102  *
 103  * FUNCTION:    usage
 104  *
 105  * PARAMETERS:  None
 106  *
 107  * RETURN:      None
 108  *
 109  * DESCRIPTION: Print a usage message
 110  *
 111  *****************************************************************************/

 113 static void
 114 usage (
 115     void)
 116 {

 118     ACPI_USAGE_HEADER ("acpiexec [options] AMLfile1 AMLfile2 ...");

 120     ACPI_OPTION ("-b \"CommandLine\"",  "Batch mode command line execution (cmd1
 121     ACPI_OPTION ("-h -?",               "Display this help message");
 122     ACPI_OPTION ("-m [Method]",         "Batch mode method execution. Default=MA
 123     printf ("\n");

 125     ACPI_OPTION ("-da",                 "Disable method abort on error");
```

```
126     ACPI_OPTION ("-di",                 "Disable execution of STA/INI methods du
127     ACPI_OPTION ("-do",                 "Disable Operation Region address simula
128     ACPI_OPTION ("-dr",                 "Disable repair of method return values"
129     ACPI_OPTION ("-dt",                 "Disable allocation tracking (performanc
130     printf ("\n");

132     ACPI_OPTION ("-ef",                 "Enable display of final memory statisti
133     ACPI_OPTION ("-ei",                 "Enable additional tests for ACPICA inte
134     ACPI_OPTION ("-em",                 "Enable Interpreter Serialized Mode");
135     ACPI_OPTION ("-es",                 "Enable Interpreter Slack Mode");
136     ACPI_OPTION ("-et",                 "Enable debug semaphore timeout");
137     printf ("\n");

139     ACPI_OPTION ("-f <Value>",          "Operation Region initialization fill va
140     ACPI_OPTION ("-r",                  "Use hardware-reduced FADT V5");
141     ACPI_OPTION ("-v",                  "Display version information");
142     ACPI_OPTION ("-vi",                 "Verbose initialization output");
143     ACPI_OPTION ("-vr",                 "Verbose region handler output");
144     ACPI_OPTION ("-x <DebugLevel>",     "Debug output level");
145 }


148 /******************************************************************************
149  *
150  * FUNCTION:    AeDoOptions
151  *
152  * PARAMETERS:  argc/argv           - Standard argc/argv
153  *
154  * RETURN:      Status
155  *
156  * DESCRIPTION: Command line option processing
157  *
158  *****************************************************************************/

160 static int
161 AeDoOptions (
162     int                     argc,
163     char                    **argv)
164 {
165     int                     j;


168     while ((j = AcpiGetopt (argc, argv, AE_SUPPORTED_OPTIONS)) != EOF) switch (j
169     {
170     case 'b':

172         if (strlen (AcpiGbl_Optarg) > (AE_BUFFER_SIZE -1))
173         {
174             printf ("**** The length of command line (%u) exceeded maximum (%u)\
175                 (UINT32) strlen (AcpiGbl_Optarg), (AE_BUFFER_SIZE -1));
176             return (-1);
177         }
178         AcpiGbl_ExecutionMode = AE_MODE_BATCH_MULTIPLE;
179         strcpy (BatchBuffer, AcpiGbl_Optarg);
180         break;

182     case 'd':

184         switch (AcpiGbl_Optarg[0])
185         {
186         case 'a':

188             AcpiGbl_IgnoreErrors = TRUE;
189             break;

191         case 'i':
```



```
193             AcpiGbl_DbOpt_ini_methods = FALSE;
194             break;

196         case 'o':

198             AcpiGbl_DbOpt_NoRegionSupport = TRUE;
199             break;

201         case 'r':

203             AcpiGbl_DisableAutoRepair = TRUE;
204             break;

206         case 't':

208             #ifdef ACPI_DBG_TRACK_ALLOCATIONS
209                 AcpiGbl_DisableMemTracking = TRUE;
210             #endif
211             break;

213         default:

215             printf ("Unknown option: -d%s\n", AcpiGbl_Optarg);
216             return (-1);
217         }
218         break;

220     case 'e':

222         switch (AcpiGbl_Optarg[0])
223         {
224         case 'f':

226             #ifdef ACPI_DBG_TRACK_ALLOCATIONS
227                 AcpiGbl_DisplayFinalMemStats = TRUE;
228             #endif
229             break;

231         case 'i':

233             AcpiGbl_DoInterfaceTests = TRUE;
234             break;

236         case 'm':

238             AcpiGbl_AllMethodsSerialized = TRUE;
239             printf ("Enabling AML Interpreter serialized mode\n");
240             break;

242         case 's':

244             AcpiGbl_EnableInterpreterSlack = TRUE;
245             printf ("Enabling AML Interpreter slack mode\n");
246             break;

248         case 't':

250             AcpiGbl_DebugTimeout = TRUE;
251             break;

253         default:

255             printf ("Unknown option: -e%s\n", AcpiGbl_Optarg);
256             return (-1);
257         }
```

```
258         break;

260     case 'f':

262         AcpiGbl_RegionFillValue = (UINT8) strtoul (AcpiGbl_Optarg, NULL, 0);
263         break;

265     case 'g':

267         AcpiGbl_DbOpt_tables = TRUE;
268         AcpiGbl_DbFilename = NULL;
269         break;

271     case 'h':
272     case '?':

274         usage();
275         return (0);

277     case 'm':

279         AcpiGbl_ExecutionMode = AE_MODE_BATCH_SINGLE;
280         switch (AcpiGbl_Optarg[0])
281         {
282         case '^':

284             strcpy (BatchBuffer, "MAIN");
285             break;

287         default:

289             strcpy (BatchBuffer, AcpiGbl_Optarg);
290             break;
291         }
292         break;

294     case 'o':

296         AcpiGbl_DbOpt_disasm = TRUE;
297         AcpiGbl_DbOpt_stats = TRUE;
298         break;

300     case 'r':

302         AcpiGbl_UseHwReducedFadt = TRUE;
303         printf ("Using ACPI 5.0 Hardware Reduced Mode via version 5 FADT\n");
304         break;

306     case 'v':

308         switch (AcpiGbl_Optarg[0])
309         {
310         case '^':  /* -v: (Version): signon already emitted, just exit */

312             exit (0);

314         case 'i':

316             AcpiDbgLevel |= ACPI_LV_INIT_NAMES;
317             break;

319         case 'r':

321             AcpiGbl_DisplayRegionAccess = TRUE;
322             break;
```



```
324         default:

326             printf ("Unknown option: -v%s\n", AcpiGbl_Optarg);
327             return (-1);
328         }
329         break;

331     case 'x':

333         AcpiDbgLevel = strtoul (AcpiGbl_Optarg, NULL, 0);
334         AcpiGbl_DbConsoleDebugLevel = AcpiDbgLevel;
335         printf ("Debug Level: 0x%8.8X\n", AcpiDbgLevel);
336         break;

338     default:

340         usage();
341         return (-1);
342     }

344     return (0);
345 }


348 /******************************************************************************
349  *
350  * FUNCTION:    main
351  *
352  * PARAMETERS:  argc, argv
353  *
354  * RETURN:      Status
355  *
356  * DESCRIPTION: Main routine for AcpiExec utility
357  *
358  *****************************************************************************/

360 int ACPI_SYSTEM_XFACE
361 main (
362     int                     argc,
363     char                    **argv)
364 {
365     ACPI_STATUS             Status;
366     UINT32                  InitFlags;
367     ACPI_TABLE_HEADER       *Table = NULL;
368     UINT32                  TableCount;
369     AE_TABLE_DESC           *TableDesc;


372     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

374     printf (ACPI_COMMON_SIGNON (ACPIEXEC_NAME));
375     if (argc < 2)
376     {
377         usage ();
378         return (0);
379     }

381     signal (SIGINT, AeCtrlCHandler);

383     /* Init globals */

385     AcpiDbgLevel = ACPI_NORMAL_DEFAULT;
386     AcpiDbgLayer = 0xFFFFFFFF;

388     /* Init ACPI and start debugger thread */
```

```
390     Status = AcpiInitializeSubsystem ();
391     AE_CHECK_OK (AcpiInitializeSubsystem, Status);
392     if (ACPI_FAILURE (Status))
393     {
394         goto ErrorExit;
395     }

397     /* Get the command line options */

399     if (AeDoOptions (argc, argv))
400     {
401         goto ErrorExit;
402     }

404     /* The remaining arguments are filenames for ACPI tables */

406     if (!argv[AcpiGbl_Optind])
407     {
408         goto EnterDebugger;
409     }

411     AcpiGbl_DbOpt_tables = TRUE;
412     TableCount = 0;

414     /* Get each of the ACPI table files on the command line */

416     while (argv[AcpiGbl_Optind])
417     {
418         /* Get one entire table */

420         Status = AcpiDbReadTableFromFile (argv[AcpiGbl_Optind], &Table);
421         if (ACPI_FAILURE (Status))
422         {
423             printf ("**** Could not get table from file %s, %s\n",
424                 argv[AcpiGbl_Optind], AcpiFormatException (Status));
425             goto ErrorExit;
426         }

428         /* Ignore non-AML tables, we can't use them. Except for an FADT */

430         if (!ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_FADT) &&
431             !AcpiUtIsAmlTable (Table))
432         {
433             ACPI_INFO ((AE_INFO,
434                 "Table [%4.4s] is not an AML table, ignoring",
435                 Table->Signature));
436             AcpiOsFree (Table);
437         }
438         else
439         {
440             /* Allocate and link a table descriptor */

442             TableDesc = AcpiOsAllocate (sizeof (AE_TABLE_DESC));
443             TableDesc->Table = Table;
444             TableDesc->Next = AeTableListHead;
445             AeTableListHead = TableDesc;

447             TableCount++;
448         }

450         AcpiGbl_Optind++;
451     }

453     /* Build a local RSDT with all tables and let ACPICA process the RSDT */

455     Status = AeBuildLocalTables (TableCount, AeTableListHead);
```



```
456     if (ACPI_FAILURE (Status))
457     {
458         goto ErrorExit;
459     }

461     Status = AeInstallTables ();
462     if (ACPI_FAILURE (Status))
463     {
464         printf ("**** Could not load ACPI tables, %s\n",
465             AcpiFormatException (Status));
466         goto EnterDebugger;
467     }

469     /*
470      * Install most of the handlers.
471      * Override some default region handlers, especially SystemMemory
472      */
473     Status = AeInstallEarlyHandlers ();
474     if (ACPI_FAILURE (Status))
475     {
476         goto EnterDebugger;
477     }

479     /* Setup initialization flags for ACPICA */

481     InitFlags = (ACPI_NO_HANDLER_INIT | ACPI_NO_ACPI_ENABLE);
482     if (!AcpiGbl_DbOpt_ini_methods)
483     {
484         InitFlags |= (ACPI_NO_DEVICE_INIT | ACPI_NO_OBJECT_INIT);
485     }

487     /*
488      * Main initialization for ACPICA subsystem
489      * TBD: Need a way to call this after the ACPI table "LOAD" command
490      */
491     Status = AcpiEnableSubsystem (InitFlags);
492     if (ACPI_FAILURE (Status))
493     {
494         printf ("**** Could not EnableSubsystem, %s\n",
495             AcpiFormatException (Status));
496         goto EnterDebugger;
497     }

499     /*
500      * Install handlers for "device driver" space IDs (EC,SMBus, etc.)
501      * and fixed event handlers
502      */
503     AeInstallLateHandlers ();

505     /* Finish the ACPICA initialization */

507     Status = AcpiInitializeObjects (InitFlags);
508     if (ACPI_FAILURE (Status))
509     {
510         printf ("**** Could not InitializeObjects, %s\n",
511             AcpiFormatException (Status));
512         goto EnterDebugger;
513     }

515     AeMiscellaneousTests ();


518 EnterDebugger:

520     /* Exit if error above and we are in one of the batch modes */
```

```
522     if (ACPI_FAILURE (Status) && (AcpiGbl_ExecutionMode > 0))
523     {
524         goto ErrorExit;
525     }

527     /* Run a batch command or enter the command loop */

529     switch (AcpiGbl_ExecutionMode)
530     {
531     default:
532     case AE_MODE_COMMAND_LOOP:

534         AcpiDbUserCommands (ACPI_DEBUGGER_COMMAND_PROMPT, NULL);
535         break;

537     case AE_MODE_BATCH_MULTIPLE:

539         AcpiDbRunBatchMode ();
540         break;

542     case AE_MODE_BATCH_SINGLE:

544         AcpiDbExecute (BatchBuffer, NULL, NULL, EX_NO_SINGLE_STEP);
545         Status = AcpiTerminate ();
546         break;
547     }

549     return (0);


552 ErrorExit:

554     (void) AcpiOsTerminate ();
555     return (-1);
556 }


559 /******************************************************************************
560  *
561  * FUNCTION:    AcpiDbRunBatchMode
562  *
563  * PARAMETERS:  BatchCommandLine    - A semicolon separated list of commands
564  *                                    to be executed.
565  *                                    Use only commas to separate elements of
566  *                                    particular command.
567  * RETURN:      Status
568  *
569  * DESCRIPTION: For each command of list separated by ';' prepare the command
570  *              buffer and pass it to AcpiDbCommandDispatch.
571  *
572  *****************************************************************************/

574 static ACPI_STATUS
575 AcpiDbRunBatchMode (
576     void)
577 {
578     ACPI_STATUS             Status;
579     char                    *Ptr = BatchBuffer;
580     char                    *Cmd = Ptr;
581     UINT8                   Run = 0;


584     AcpiGbl_MethodExecuting = FALSE;
585     AcpiGbl_StepToNextCall = FALSE;

587     while (*Ptr)
```



```
588     {
589         if (*Ptr == ',')
590         {
591             /* Convert commas to spaces */
592             *Ptr = ' ';
593         }
594         else if (*Ptr == ';')
595         {
596             *Ptr = '\0';
597             Run = 1;
598         }

600         Ptr++;

602         if (Run || (*Ptr == '\0'))
603         {
604             (void) AcpiDbCommandDispatch (Cmd, NULL, NULL);
605             Run = 0;
606             Cmd = Ptr;
607         }
608     }

610     Status = AcpiTerminate ();
611     return (Status);
612 }
```

```
**********************************************************
   15669 Thu Dec 26 13:50:09 2013
new/usr/src/common/acpica/tools/acpiexec/aetables.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aetables - ACPI table setup/install for acpiexec utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "aecommon.h"
#include "aetables.h"

#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("aetables")

/* Local prototypes */

void
AeTableOverride (
    ACPI_TABLE_HEADER       *ExistingTable,
    ACPI_TABLE_HEADER       **NewTable);

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);
```



```
/* User table (DSDT) */

static ACPI_TABLE_HEADER        *DsdtToInstallOverride;

/* Non-AML tables that are constructed locally and installed */

static ACPI_TABLE_RSDP          LocalRSDP;
static ACPI_TABLE_FACS          LocalFACS;
static ACPI_TABLE_HEADER        LocalTEST;
static ACPI_TABLE_HEADER        LocalBADTABLE;

/*
 * We need a local FADT so that the hardware subcomponent will function,
 * even though the underlying OSD HW access functions don't do anything.
 */
static ACPI_TABLE_FADT          LocalFADT;

/*
 * Use XSDT so that both 32- and 64-bit versions of this utility will
 * function automatically.
 */
static ACPI_TABLE_XSDT          *LocalXSDT;

#define BASE_XSDT_TABLES        10
#define BASE_XSDT_SIZE          (sizeof (ACPI_TABLE_XSDT) + \
                                    ((BASE_XSDT_TABLES -1) * sizeof (UINT64)))

#define ACPI_MAX_INIT_TABLES    (32)
static ACPI_TABLE_DESC          Tables[ACPI_MAX_INIT_TABLES];


/******************************************************************************
 *
 * FUNCTION:    AeTableOverride
 *
 * DESCRIPTION: Local implementation of AcpiOsTableOverride.
 *              Exercise the override mechanism
 *
 *****************************************************************************/

void
AeTableOverride (
    ACPI_TABLE_HEADER       *ExistingTable,
    ACPI_TABLE_HEADER       **NewTable)
{

    /* This code exercises the table override mechanism in the core */

    if (ACPI_COMPARE_NAME (ExistingTable->Signature, ACPI_SIG_DSDT))
    {
        *NewTable = DsdtToInstallOverride;
    }

    /* This code tests override of dynamically loaded tables */

    else if (ACPI_COMPARE_NAME (ExistingTable->Signature, "OEM9"))
    {
        *NewTable = ACPI_CAST_PTR (ACPI_TABLE_HEADER, Ssdt3Code);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AeBuildLocalTables
```

```
126  *
127  * PARAMETERS:  TableCount      - Number of tables on the command line
128  *              TableList       - List of actual tables from files
129  *
130  * RETURN:      Status
131  *
132  * DESCRIPTION: Build a complete ACPI table chain, with a local RSDP, XSDT,
133  *              FADT, and several other test tables.
134  *
135  *****************************************************************************/

137 ACPI_STATUS
138 AeBuildLocalTables (
139     UINT32                  TableCount,
140     AE_TABLE_DESC           *TableList)
141 {
142     ACPI_PHYSICAL_ADDRESS   DsdtAddress = 0;
143     UINT32                  XsdtSize;
144     AE_TABLE_DESC           *NextTable;
145     UINT32                  NextIndex;
146     ACPI_TABLE_FADT         *ExternalFadt = NULL;


149     /*
150      * Update the table count. For DSDT, it is not put into the XSDT. For
151      * FADT, this is already accounted for since we usually install a
152      * local FADT.
153      */
154     NextTable = TableList;
155     while (NextTable)
156     {
157         if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT) ||
158             ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
159         {
160             TableCount--;
161         }
162         NextTable = NextTable->Next;
163     }

165     XsdtSize = BASE_XSDT_SIZE + (TableCount * sizeof (UINT64));

167     /* Build an XSDT */

169     LocalXSDT = AcpiOsAllocate (XsdtSize);
170     if (!LocalXSDT)
171     {
172         return (AE_NO_MEMORY);
173     }

175     ACPI_MEMSET (LocalXSDT, 0, XsdtSize);
176     ACPI_MOVE_NAME (LocalXSDT->Header.Signature, ACPI_SIG_XSDT);
177     LocalXSDT->Header.Length = XsdtSize;
178     LocalXSDT->Header.Revision = 1;

180     LocalXSDT->TableOffsetEntry[0] = ACPI_PTR_TO_PHYSADDR (&LocalTEST);
181     LocalXSDT->TableOffsetEntry[1] = ACPI_PTR_TO_PHYSADDR (&LocalBADTABLE);
182     LocalXSDT->TableOffsetEntry[2] = ACPI_PTR_TO_PHYSADDR (&LocalFADT);

184     /* Install two SSDTs to test multiple table support */

186     LocalXSDT->TableOffsetEntry[3] = ACPI_PTR_TO_PHYSADDR (&Ssdt1Code);
187     LocalXSDT->TableOffsetEntry[4] = ACPI_PTR_TO_PHYSADDR (&Ssdt2Code);

189     /* Install the OEM1 table to test LoadTable */

191     LocalXSDT->TableOffsetEntry[5] = ACPI_PTR_TO_PHYSADDR (&Oem1Code);
```



```
193     /* Install the OEMx table to test LoadTable */

195     LocalXSDT->TableOffsetEntry[6] = ACPI_PTR_TO_PHYSADDR (&OemxCode);

197      /* Install the ECDT table to test _REG */

199     LocalXSDT->TableOffsetEntry[7] = ACPI_PTR_TO_PHYSADDR (&EcdtCode);

201     /* Install two UEFIs to test multiple table support */

203     LocalXSDT->TableOffsetEntry[8] = ACPI_PTR_TO_PHYSADDR (&Uefi1Code);
204     LocalXSDT->TableOffsetEntry[9] = ACPI_PTR_TO_PHYSADDR (&Uefi2Code);

206     /*
207      * Install the user tables. The DSDT must be installed in the FADT.
208      * All other tables are installed directly into the XSDT.
209      */
210     NextIndex = BASE_XSDT_TABLES;
211     NextTable = TableList;
212     while (NextTable)
213     {
214         /*
215          * Incoming DSDT or FADT are special cases. All other tables are
216          * just immediately installed into the XSDT.
217          */
218         if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT))
219         {
220             if (DsdtAddress)
221             {
222                 printf ("Already found a DSDT, only one allowed\n");
223                 return (AE_ALREADY_EXISTS);
224             }

226             /* The incoming user table is a DSDT */

228             DsdtAddress = ACPI_PTR_TO_PHYSADDR (&DsdtCode);
229             DsdtToInstallOverride = NextTable->Table;
230         }
231         else if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
232         {
233             ExternalFadt = ACPI_CAST_PTR (ACPI_TABLE_FADT, NextTable->Table);
234             LocalXSDT->TableOffsetEntry[2] = ACPI_PTR_TO_PHYSADDR (NextTable->Ta
235         }
236         else
237         {
238             /* Install the table in the XSDT */

240             LocalXSDT->TableOffsetEntry[NextIndex] = ACPI_PTR_TO_PHYSADDR (NextT
241             NextIndex++;
242         }

244         NextTable = NextTable->Next;
245     }

247     /* Build an RSDP */

249     ACPI_MEMSET (&LocalRSDP, 0, sizeof (ACPI_TABLE_RSDP));
250     ACPI_MAKE_RSDP_SIG (LocalRSDP.Signature);
251     ACPI_MEMCPY (LocalRSDP.OemId, "I_TEST", 6);
252     LocalRSDP.Revision = 2;
253     LocalRSDP.XsdtPhysicalAddress = ACPI_PTR_TO_PHYSADDR (LocalXSDT);
254     LocalRSDP.Length = sizeof (ACPI_TABLE_XSDT);

256     /* Set checksums for both XSDT and RSDP */
```

```
    LocalXSDT->Header.Checksum = (UINT8) -AcpiTbChecksum (
        (void *) LocalXSDT, LocalXSDT->Header.Length);
    LocalRSDP.Checksum = (UINT8) -AcpiTbChecksum (
        (void *) &LocalRSDP, ACPI_RSDP_CHECKSUM_LENGTH);

    if (!DsdtAddress)
    {
        /* Use the local DSDT because incoming table(s) are all SSDT(s) */

        DsdtAddress = ACPI_PTR_TO_PHYSADDR (LocalDsdtCode);
        DsdtToInstallOverride = ACPI_CAST_PTR (ACPI_TABLE_HEADER, LocalDsdtCode)
    }

    if (ExternalFadt)
    {
        /*
         * Use the external FADT, but we must update the DSDT/FACS addresses
         * as well as the checksum
         */
        ExternalFadt->Dsdt = DsdtAddress;
        if (!AcpiGbl_ReducedHardware)
        {
            ExternalFadt->Facs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
        }

        if (ExternalFadt->Header.Length > ACPI_PTR_DIFF (&ExternalFadt->XDsdt, E
        {
            ExternalFadt->XDsdt = DsdtAddress;

            if (!AcpiGbl_ReducedHardware)
            {
                ExternalFadt->XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
            }
        }

        /* Complete the FADT with the checksum */

        ExternalFadt->Header.Checksum = 0;
        ExternalFadt->Header.Checksum = (UINT8) -AcpiTbChecksum (
            (void *) ExternalFadt, ExternalFadt->Header.Length);
    }
    else if (AcpiGbl_UseHwReducedFadt)
    {
        ACPI_MEMCPY (&LocalFADT, HwReducedFadtCode, sizeof (ACPI_TABLE_FADT));
        LocalFADT.Dsdt = DsdtAddress;
        LocalFADT.XDsdt = DsdtAddress;

        LocalFADT.Header.Checksum = 0;
        LocalFADT.Header.Checksum = (UINT8) -AcpiTbChecksum (
            (void *) &LocalFADT, LocalFADT.Header.Length);
    }
    else
    {
        /*
         * Build a local FADT so we can test the hardware/event init
         */
        ACPI_MEMSET (&LocalFADT, 0, sizeof (ACPI_TABLE_FADT));
        ACPI_MOVE_NAME (LocalFADT.Header.Signature, ACPI_SIG_FADT);

        /* Setup FADT header and DSDT/FACS addresses */

        LocalFADT.Dsdt = 0;
        LocalFADT.Facs = 0;

        LocalFADT.XDsdt = DsdtAddress;
        LocalFADT.XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
```



```
        LocalFADT.Header.Revision = 3;
        LocalFADT.Header.Length = sizeof (ACPI_TABLE_FADT);

        /* Miscellaneous FADT fields */

        LocalFADT.Gpe0BlockLength = 16;
        LocalFADT.Gpe0Block = 0x00001234;

        LocalFADT.Gpe1BlockLength = 6;
        LocalFADT.Gpe1Block = 0x00005678;
        LocalFADT.Gpe1Base = 96;

        LocalFADT.Pm1EventLength = 4;
        LocalFADT.Pm1aEventBlock = 0x00001aaa;
        LocalFADT.Pm1bEventBlock = 0x00001bbb;

        LocalFADT.Pm1ControlLength = 2;
        LocalFADT.Pm1aControlBlock = 0xB0;

        LocalFADT.PmTimerLength = 4;
        LocalFADT.PmTimerBlock = 0xA0;

        LocalFADT.Pm2ControlBlock = 0xC0;
        LocalFADT.Pm2ControlLength = 1;

        /* Setup one example X-64 field */

        LocalFADT.XPm1bEventBlock.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;
        LocalFADT.XPm1bEventBlock.Address = LocalFADT.Pm1bEventBlock;
        LocalFADT.XPm1bEventBlock.BitWidth = (UINT8) ACPI_MUL_8 (LocalFADT.Pm1Ev

        /* Complete the FADT with the checksum */

        LocalFADT.Header.Checksum = 0;
        LocalFADT.Header.Checksum = (UINT8) -AcpiTbChecksum (
            (void *) &LocalFADT, LocalFADT.Header.Length);
    }

    /* Build a FACS */

    ACPI_MEMSET (&LocalFACS, 0, sizeof (ACPI_TABLE_FACS));
    ACPI_MOVE_NAME (LocalFACS.Signature, ACPI_SIG_FACS);

    LocalFACS.Length = sizeof (ACPI_TABLE_FACS);
    LocalFACS.GlobalLock = 0x11AA0011;

    /*
     * Build a fake table [TEST] so that we make sure that the
     * ACPICA core ignores it
     */
    ACPI_MEMSET (&LocalTEST, 0, sizeof (ACPI_TABLE_HEADER));
    ACPI_MOVE_NAME (LocalTEST.Signature, "TEST");

    LocalTEST.Revision = 1;
    LocalTEST.Length = sizeof (ACPI_TABLE_HEADER);
    LocalTEST.Checksum = (UINT8) -AcpiTbChecksum (
        (void *) &LocalTEST, LocalTEST.Length);

    /*
     * Build a fake table with a bad signature [BAD!] so that we make
     * sure that the ACPICA core ignores it
     */
    ACPI_MEMSET (&LocalBADTABLE, 0, sizeof (ACPI_TABLE_HEADER));
    ACPI_MOVE_NAME (LocalBADTABLE.Signature, "BAD!");
```

```
390     LocalBADTABLE.Revision = 1;
391     LocalBADTABLE.Length = sizeof (ACPI_TABLE_HEADER);
392     LocalBADTABLE.Checksum = (UINT8) -AcpiTbChecksum (
393         (void *) &LocalBADTABLE, LocalBADTABLE.Length);

395     return (AE_OK);
396 }


399 /******************************************************************************
400  *
401  * FUNCTION:    AeInstallTables
402  *
403  * PARAMETERS:  None
404  *
405  * RETURN:      Status
406  *
407  * DESCRIPTION: Install the various ACPI tables
408  *
409  *****************************************************************************/

411 ACPI_STATUS
412 AeInstallTables (
413     void)
414 {
415     ACPI_STATUS             Status;
416     ACPI_TABLE_HEADER       Header;
417     ACPI_TABLE_HEADER       *Table;


420     Status = AcpiInitializeTables (Tables, ACPI_MAX_INIT_TABLES, TRUE);
421     AE_CHECK_OK (AcpiInitializeTables, Status);

423     Status = AcpiReallocateRootTable ();
424     AE_CHECK_OK (AcpiReallocateRootTable, Status);

426     Status = AcpiLoadTables ();
427     AE_CHECK_OK (AcpiLoadTables, Status);

429     /*
430      * Test run-time control method installation. Do it twice to test code
431      * for an existing name.
432      */
433     Status = AcpiInstallMethod (MethodCode);
434     if (ACPI_FAILURE (Status))
435     {
436         AcpiOsPrintf ("%s, Could not install method\n",
437             AcpiFormatException (Status));
438     }

440     Status = AcpiInstallMethod (MethodCode);
441     if (ACPI_FAILURE (Status))
442     {
443         AcpiOsPrintf ("%s, Could not install method\n",
444             AcpiFormatException (Status));
445     }

447     /* Test multiple table/UEFI support. First, get the headers */

449     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 1, &Header);
450     AE_CHECK_OK (AcpiGetTableHeader, Status);

452     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 2, &Header);
453     AE_CHECK_OK (AcpiGetTableHeader, Status);

455     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 3, &Header);
```



```
456     AE_CHECK_STATUS (AcpiGetTableHeader, Status, AE_NOT_FOUND);

458     /* Now get the actual tables */

460     Status = AcpiGetTable (ACPI_SIG_UEFI, 1, &Table);
461     AE_CHECK_OK (AcpiGetTable, Status);

463     Status = AcpiGetTable (ACPI_SIG_UEFI, 2, &Table);
464     AE_CHECK_OK (AcpiGetTable, Status);

466     Status = AcpiGetTable (ACPI_SIG_UEFI, 3, &Table);
467     AE_CHECK_STATUS (AcpiGetTable, Status, AE_NOT_FOUND);

469     return (AE_OK);
470 }


473 /******************************************************************************
474  *
475  * FUNCTION:    AeLocalGetRootPointer
476  *
477  * PARAMETERS:  Flags       - not used
478  *              Address     - Where the root pointer is returned
479  *
480  * RETURN:      Status
481  *
482  * DESCRIPTION: Return a local RSDP, used to dynamically load tables via the
483  *              standard ACPI mechanism.
484  *
485  *****************************************************************************/

487 ACPI_PHYSICAL_ADDRESS
488 AeLocalGetRootPointer (
489     void)
490 {

492     return ((ACPI_PHYSICAL_ADDRESS) &LocalRSDP);
493 }
```

```
**********************************************************
   22354 Thu Dec 26 13:50:09 2013
new/usr/src/common/acpica/tools/acpiexec/aetables.h
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: aetables.h - Precompiled AML ACPI tables for acpiexec
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __AETABLES_H__
#define __AETABLES_H__


/*
 * Miscellaneous pre-compiled AML ACPI tables to be installed
 */

/* Default DSDT. This will be replaced with the input DSDT */

static unsigned char DsdtCode[] =
{
    0x44,0x53,0x44,0x54,0x24,0x00,0x00,0x00,  /* 00000000    "DSDT$..." */
    0x02,0x6F,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    ".oIntel." */
    0x4E,0x75,0x6C,0x6C,0x44,0x53,0x44,0x54,  /* 00000010    "NullDSDT" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x04,0x12,0x08,0x20,
};

static unsigned char LocalDsdtCode[] =
{
    0x44,0x53,0x44,0x54,0x24,0x00,0x00,0x00,  /* 00000000    "DSDT$..." */
    0x02,0x2C,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    ".,Intel." */
    0x4C,0x6F,0x63,0x61,0x6C,0x00,0x00,0x00,  /* 00000010    "Local..." */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x30,0x07,0x09,0x20,
};

/* Several example SSDTs */

/* SSDT1 is used by ASLTS; if changed here, must also be changed in dtregions.as

static unsigned char Ssdt1Code[] = /* Has method _T98 */
{
    0x53,0x53,0x44,0x54,0x3E,0x00,0x00,0x00,  /* 00000000    "SSDT>..." */
    0x02,0x08,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    "..Intel." */
    0x73,0x73,0x64,0x74,0x31,0x00,0x00,0x00,  /* 00000010    "ssdt1..." */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x20,0x06,0x12,0x20,0x14,0x19,0x5F,0x54,  /* 00000020    " .. .._T" */
    0x39,0x38,0x01,0x70,0x0D,0x53,0x53,0x44,  /* 00000028    "98.p.SSD" */
    0x54,0x31,0x20,0x2D,0x20,0x5F,0x54,0x39,  /* 00000030    "T1 - _T9" */
    0x38,0x00,0x5B,0x31,0xA4,0x00             /* 00000038    "8.[1.."   */
};

unsigned char Ssdt2Code[] = /* Has method _T99 */
{
    0x53,0x53,0x44,0x54,0x3E,0x00,0x00,0x00,  /* 00000000    "SSDT>..." */
    0x02,0xFE,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    "..Intel." */
    0x73,0x73,0x64,0x74,0x32,0x00,0x00,0x00,  /* 00000010    "ssdt2..." */
    0x02,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x20,0x06,0x12,0x20,0x14,0x19,0x5F,0x54,  /* 00000020    " .. .._T" */
    0x39,0x39,0x06,0x70,0x0D,0x53,0x53,0x44,  /* 00000028    "99.p.SSD" */
    0x54,0x32,0x20,0x2D,0x20,0x5F,0x54,0x39,  /* 00000030    "T2 - _T9" */
    0x39,0x00,0x5B,0x31,0xA4,0x00             /* 00000038    "9.[1.."   */
};

unsigned char Ssdt3Code[] =     /* OEM9: Has method _T97 */
{
    0x4F,0x45,0x4D,0x39,0x30,0x00,0x00,0x00,  /* 00000000    "OEM10..." */
    0x01,0xDD,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    "..Intel." */
    0x4D,0x61,0x6E,0x79,0x00,0x00,0x00,0x00,  /* 00000010    "Many...." */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x24,0x04,0x03,0x20,0x14,0x0B,0x5F,0x54,  /* 00000020    "$.. .._T" */
    0x39,0x37,0x00,0x70,0x0A,0x04,0x60,0xA4,  /* 00000028    "97.p..`." */
};

unsigned char Ssdt4Code[] =     /* Has method _T96 */
{
    0x53,0x53,0x44,0x54,0x2D,0x00,0x00,0x00,  /* 00000000    "SSDT-..." */
    0x02,0x2B,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    ".+Intel." */
    0x73,0x73,0x64,0x74,0x34,0x00,0x00,0x00,  /* 00000010    "ssdt4..." */
    0x04,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x20,0x06,0x12,0x20,0x14,0x08,0x5F,0x54,  /* 00000020    " .. .._T" */
    0x39,0x36,0x05,0xA4,0x00                  /* 00000028    "96..."    */
};

/* "Hardware-Reduced" ACPI 5.0 FADT (No FACS, no ACPI hardware) */

unsigned char HwReducedFadtCode[] =
{
    0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00,  /* 00000000    "FACP...." */
    0x05,0x8C,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
```

```
126     0x41,0x43,0x50,0x49,0x35,0x30,0x20,0x20,  /* 00000010    "ACPI50  " */
127     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
128     0x13,0x04,0x11,0x20,0x00,0x00,0x00,0x00,  /* 00000020    "... ...." */
129     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
130     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
131     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
132     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
133     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000048    "........" */
134     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
135     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000058    "........" */
136     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
137     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
138     0x00,0x00,0x78,0x00,0x01,0x08,0x00,0x01,  /* 00000070    "..x....." */
139     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
140     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000080    "........" */
141     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000088    "........" */
142     0x00,0x00,0x00,0x00,0x01,0x20,0x00,0x02,  /* 00000090    "..... .." */
143     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
144     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
145     0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x02,  /* 000000A8    "........" */
146     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B0    "........" */
147     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
148     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x00,  /* 000000C0    "........" */
149     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
150     0x01,0x20,0x00,0x03,0x00,0x00,0x00,0x00,  /* 000000D0    ". ......" */
151     0x00,0x00,0x00,0x00,0x01,0x80,0x00,0x01,  /* 000000D8    "........" */
152     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
153     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
154     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01,  /* 000000F0    "........" */
155     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
156     0x01,0x08,0x00,0x01,0x01,0x00,0x00,0x00,  /* 00000100    "........" */
157     0x00,0x00,0x00,0x00                       /* 00000108    "........" */
158 };

160 /* Example OEM table */

162 static unsigned char Oem1Code[] =
163 {
164     0x4F,0x45,0x4D,0x31,0x38,0x00,0x00,0x00,  /* 00000000    "OEM18..." */
165     0x01,0x4B,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    ".KIntel." */
166     0x4D,0x61,0x6E,0x79,0x00,0x00,0x00,0x00,  /* 00000010    "Many...." */
167     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
168     0x18,0x09,0x03,0x20,0x08,0x5F,0x58,0x54,  /* 00000020    "... ._XT" */
169     0x32,0x0A,0x04,0x14,0x0C,0x5F,0x58,0x54,  /* 00000028    "2...._XT" */
170     0x31,0x00,0x70,0x01,0x5F,0x58,0x54,0x32,  /* 00000030    "1.p._XT2" */
171 };

173 /* ASL source for this table is at the end of this file */

175 static unsigned char OemxCode[] =
176 {
177     0x4F,0x45,0x4D,0x58,0xB0,0x00,0x00,0x00,  /* 00000000    "OEMX...." */
178     0x02,0x54,0x4D,0x79,0x4F,0x45,0x4D,0x00,  /* 00000008    ".TMyOEM." */
179     0x54,0x65,0x73,0x74,0x00,0x00,0x00,0x00,  /* 00000010    "Test...." */
180     0x32,0x04,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "2...INTL" */
181     0x31,0x03,0x10,0x20,0x14,0x1D,0x5F,0x49,  /* 00000020    "1.. .._I" */
182     0x4E,0x49,0x00,0x70,0x0D,0x54,0x61,0x62,  /* 00000028    "NI.p.Tab" */
183     0x6C,0x65,0x20,0x4F,0x45,0x4D,0x58,0x20,  /* 00000030    "le OEMX " */
184     0x72,0x75,0x6E,0x6E,0x69,0x6E,0x67,0x00,  /* 00000038    "running." */
185     0x5B,0x31,0x10,0x22,0x5C,0x5F,0x47,0x50,  /* 00000040    "[1."\_GP" */
186     0x45,0x14,0x06,0x5F,0x45,0x30,0x37,0x00,  /* 00000048    "E._E07." */
187     0x14,0x06,0x5F,0x45,0x32,0x32,0x00,0x14,  /* 00000050    ".._E22.." */
188     0x06,0x5F,0x4C,0x33,0x31,0x00,0x14,0x06,  /* 00000058    "._L31..." */
189     0x5F,0x4C,0x36,0x36,0x00,0x5B,0x82,0x10,  /* 00000060    "_L66.[.." */
190     0x4F,0x45,0x4D,0x31,0x08,0x5F,0x50,0x52,  /* 00000068    "OEM1._PR" */
191     0x57,0x12,0x05,0x02,0x0A,0x07,0x00,0x5B,  /* 00000070    "W......[" */
```



```
192     0x82,0x10,0x4F,0x45,0x4D,0x32,0x08,0x5F,  /* 00000078    "..OEM2._" */
193     0x50,0x52,0x57,0x12,0x05,0x02,0x0A,0x66,  /* 00000080    "PRW....f" */
194     0x00,0x10,0x26,0x5C,0x47,0x50,0x45,0x32,  /* 00000088    "..&\GPE2" */
195     0x14,0x06,0x5F,0x4C,0x30,0x31,0x00,0x14,  /* 00000090    ".._L01.." */
196     0x06,0x5F,0x45,0x30,0x37,0x00,0x08,0x5F,  /* 00000098    "._E07.._" */
197     0x50,0x52,0x57,0x12,0x0C,0x02,0x12,0x08,  /* 000000A0    "PRW....." */
198     0x02,0x5C,0x47,0x50,0x45,0x32,0x01,0x00   /* 000000A8    ".\GPE2.." */
199 };

201 /* Example ECDT */

203 unsigned char EcdtCode[] =
204 {
205     0x45,0x43,0x44,0x54,0x4E,0x00,0x00,0x00,  /* 00000000    "ECDTN..." */
206     0x01,0x94,0x20,0x49,0x6E,0x74,0x65,0x6C,  /* 00000008    ".. Intel" */
207     0x54,0x65,0x6D,0x70,0x6C,0x61,0x74,0x65,  /* 00000010    "Template" */
208     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
209     0x16,0x03,0x11,0x20,0x01,0x08,0x00,0x00,  /* 00000020    "... ...." */
210     0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "f......." */
211     0x01,0x08,0x00,0x00,0x62,0x00,0x00,0x00,  /* 00000030    "....b..." */
212     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
213     0x09,0x5C,0x5F,0x53,0x42,0x2E,0x50,0x43,  /* 00000040    ".\_SB.PC" */
214     0x49,0x30,0x2E,0x45,0x43,0x00             /* 00000048    "I0.EC."   */
215 };

217 /* Test for multiple UEFI tables */

219 unsigned char Uefi1Code[] =
220 {
221     0x55,0x45,0x46,0x49,0x36,0x00,0x00,0x00,  /* 00000000    "UEFI6..." */
222     0x01,0x6E,0x20,0x49,0x6E,0x74,0x65,0x6C,  /* 00000008    ".n Intel" */
223     0x20,0x20,0x20,0x55,0x45,0x46,0x49,0x31,  /* 00000010    "   UEFI1" */
224     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
225     0x23,0x08,0x13,0x20,0x06,0x07,0x08,0x09,  /* 00000020    "#.. ...." */
226     0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,  /* 00000028    "........" */
227     0x0C,0x0D,0x0E,0x0F,0x36,0x00             /* 00000030    "....6."   */
228 };

230 unsigned char Uefi2Code[] =
231 {
232     0x55,0x45,0x46,0x49,0xAA,0x00,0x00,0x00,  /* 00000000    "UEFI...." */
233     0x01,0xE0,0x20,0x49,0x6E,0x74,0x65,0x6C,  /* 00000008    ".. Intel" */
234     0x20,0x20,0x20,0x55,0x45,0x46,0x49,0x32,  /* 00000010    "   UEFI2" */
235     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
236     0x23,0x08,0x13,0x20,0x67,0x45,0x23,0x01,  /* 00000020    "#.. gE#." */
237     0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,  /* 00000028    "........" */
238     0x0C,0x0D,0x0E,0x0F,0x36,0x00,0x04,0x19,  /* 00000030    "....6..." */
239     0x00,0x56,0x34,0xF2,0x04,0x03,0x02,0x01,  /* 00000038    ".V4....." */
240     0x77,0x66,0x55,0x44,0x33,0x22,0x11,0x1E,  /* 00000040    "wfUD3".." */
241     0x1C,0x1F,0x14,0x10,0x0C,0x08,0x04,0xAB,  /* 00000048    "........" */
242     0x54,0x68,0x69,0x73,0x20,0x69,0x73,0x20,  /* 00000050    "This is " */
243     0x61,0x20,0x73,0x74,0x72,0x69,0x6E,0x67,  /* 00000058    "a string" */
244     0x00,0x5C,0x50,0x43,0x49,0x30,0x5C,0x41,  /* 00000060    ".\PCI0\A" */
245     0x42,0x43,0x44,0x00,0x36,0x00,0x55,0x00,  /* 00000068    "BCD.6.U." */
246     0x6E,0x00,0x69,0x00,0x63,0x00,0x6F,0x00,  /* 00000070    "n.i.c.o." */
247     0x64,0x00,0x65,0x00,0x20,0x00,0x53,0x00,  /* 00000078    "d.e. .S." */
248     0x74,0x00,0x72,0x00,0x69,0x00,0x6E,0x00,  /* 00000080    "t.r.i.n." */
249     0x67,0x00,0x00,0x00,0x58,0x5B,0x00,0x00,  /* 00000088    "g...X[.." */
250     0x00,0x00,0x00,0x00,0x41,0x42,0x43,0x44,  /* 00000090    "....ABCD" */
251     0x45,0x00,0x00,0x01,0x02,0x03,0x04,0x05,  /* 00000098    "E......." */
252     0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D,  /* 000000A0    "........" */
253     0x0E,0x0F                                 /* 000000A8    ".."       */
254 };


257 /*
```

```
 * Example installable control method
 *
 * DefinitionBlock ("", "DSDT", 2, "Intel", "MTHDTEST", 0x20090512)
 * {
 *     Method (\_SI_._T97, 1, Serialized)
 *     {
 *         Store ("Example installed method", Debug)
 *         Store (Arg0, Debug)
 *         Return ()
 *     }
 * }
 *
 * Compiled byte code below.
 */
static unsigned char MethodCode[] =
{
    0x44,0x53,0x44,0x54,0x53,0x00,0x00,0x00,  /* 00000000    "DSDTS..." */
    0x02,0xF9,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    "..Intel." */
    0x4D,0x54,0x48,0x44,0x54,0x45,0x53,0x54,  /* 00000010    "MTHDTEST" */
    0x12,0x05,0x09,0x20,0x49,0x4E,0x54,0x4C,  /* 00000018    "... INTL" */
    0x22,0x04,0x09,0x20,0x14,0x2E,0x2E,0x5F,  /* 00000020    "".. ..._" */
    0x54,0x49,0x5F,0x5F,0x54,0x39,0x37,0x09,  /* 00000028    "SI__T97." */
    0x70,0x0D,0x45,0x78,0x61,0x6D,0x70,0x6C,  /* 00000030    "p.Exampl" */
    0x65,0x20,0x69,0x6E,0x73,0x74,0x61,0x6C,  /* 00000038    "e instal" */
    0x6C,0x65,0x64,0x20,0x6D,0x65,0x74,0x68,  /* 00000040    "led meth" */
    0x6F,0x64,0x00,0x5B,0x31,0x70,0x68,0x5B,  /* 00000048    "od.[1ph[" */
    0x31,0xA4,0x00,
};


#if 0
/*****************************************************************************
 *
 * DESCRIPTION: ASL tables that are used in RSDT/XSDT, also used to test
 *              Load/LoadTable operators.
 *
 ****************************************************************************/

DefinitionBlock ("", "OEMX", 2, "MyOEM", "Test", 0x00000432)
{
    External (GPE2, DeviceObj)

    Method (_INI)
    {
        Store ("Table OEMX running", Debug)
    }

    Scope (\_GPE)
    {
        Method (_E07) {}
        Method (_E22) {}
        Method (_L31) {}
        Method (_L66) {}
    }

    Device (OEM1)
    {
        Name (_PRW, Package(){7,0})
    }
    Device (OEM2)
    {
        Name (_PRW, Package(){0x66,0})
    }

    Scope (\GPE2)
    {
```



```
        Method (_L01) {}
        Method (_E07) {}

        Name (_PRW, Package() {Package() {\GPE2, 1}, 0})
    }
}

/* Parent gr.asl file */

DefinitionBlock ("", "DSDT", 2, "Intel", "Many", 0x00000001)
{
    Name (BUF1, Buffer()
    {
        0x4F,0x45,0x4D,0x58,0xB0,0x00,0x00,0x00,  /* 00000000    "OEMX...." */
        0x02,0x54,0x4D,0x79,0x4F,0x45,0x4D,0x00,  /* 00000008    ".TMyOEM." */
        0x54,0x65,0x73,0x74,0x00,0x00,0x00,0x00,  /* 00000010    "Test...." */
        0x32,0x04,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "2...INTL" */
        0x31,0x03,0x10,0x20,0x14,0x1D,0x5F,0x49,  /* 00000020    "1.. .._I" */
        0x4E,0x49,0x00,0x70,0x0D,0x54,0x61,0x62,  /* 00000028    "NI.p.Tab" */
        0x6C,0x65,0x20,0x4F,0x45,0x4D,0x58,0x20,  /* 00000030    "le OEMX " */
        0x72,0x75,0x6E,0x6E,0x69,0x6E,0x67,0x00,  /* 00000038    "running." */
        0x5B,0x31,0x10,0x22,0x5C,0x5F,0x47,0x50,  /* 00000040    "[1."\_GP" */
        0x45,0x14,0x06,0x5F,0x45,0x30,0x37,0x00,  /* 00000048    "E.._E07." */
        0x14,0x06,0x5F,0x45,0x32,0x32,0x00,0x14,  /* 00000050    ".._E22.." */
        0x06,0x5F,0x4C,0x33,0x31,0x00,0x14,0x06,  /* 00000058    "._L31..." */
        0x5F,0x4C,0x36,0x36,0x00,0x5B,0x82,0x10,  /* 00000060    "_L66.[.." */
        0x4F,0x45,0x4D,0x31,0x08,0x5F,0x50,0x52,  /* 00000068    "OEM1._PR" */
        0x57,0x12,0x05,0x02,0x0A,0x07,0x00,0x5B,  /* 00000070    "W......[" */
        0x82,0x10,0x4F,0x45,0x4D,0x32,0x08,0x5F,  /* 00000078    "..OEM2._" */
        0x50,0x52,0x57,0x12,0x05,0x02,0x0A,0x66,  /* 00000080    "PRW....f" */
        0x00,0x10,0x26,0x5C,0x47,0x50,0x45,0x32,  /* 00000088    "..&\GPE2" */
        0x14,0x06,0x5F,0x4C,0x30,0x31,0x00,0x14,  /* 00000090    ".._L01.." */
        0x06,0x5F,0x45,0x30,0x37,0x00,0x08,0x5F,  /* 00000098    "._E07.._" */
        0x50,0x52,0x57,0x12,0x0C,0x02,0x12,0x08,  /* 000000A0    "PRW....." */
        0x02,0x5C,0x47,0x50,0x45,0x32,0x01,0x00   /* 000000A8    ".\GPE2.." */
    })

    Name (HNDL, 0)
    Method (LD)
    {
        Load (BUF1, HNDL)
        Store ("Load operator, handle:", Debug)
        Store (HNDL, Debug)
    }

    Method (MAIN, 0, NotSerialized)
    {
        Store ("Loading OEMX table", Debug)
        Store (LoadTable ("OEMX", "MyOEM", "Test"), Debug)
    }

    Scope (\_GPE)
    {
        Method (_L08) {}
        Method (_E08) {}
        Method (_L0B) {}
    }

    Device (DEV0)
    {
        Name (_PRW, Package() {0x11, 0})
    }

    Device (\GPE2)
    {
        Method (_L00) {}
```

```
390     }
391 }

393 /* SSDT1 */

395 DefinitionBlock ("ssdt1.aml", "SSDT", 2, "Intel", "ssdt1", 0x00000001)
396 {
397     Method (_T98, 1, NotSerialized)
398     {
399         Store ("SSDT1 - _T98", Debug)
400         Return (Zero)
401     }
402 }

404 /* SSDT2 */

406 DefinitionBlock ("ssdt2.aml", "SSDT", 2, "Intel", "ssdt2", 0x00000002)
407 {
408     Method (_T99, 6, NotSerialized)
409     {
410         Store ("SSDT2 - _T99", Debug)
411         Return (Zero)
412     }
413 }

415 /* SSDT4 */

417 DefinitionBlock ("ssdt4.aml", "SSDT", 2, "Intel", "ssdt4", 0x00000004)
418 {
419     Method (_T96, 5, NotSerialized)
420     {
421         Return (Zero)
422     }
423 }

425 /* Example ECDT */

427 [000h 0000   4]                    Signature : "ECDT"    /* Embedded Controller
428 [004h 0004   4]                 Table Length : 0000004E
429 [008h 0008   1]                     Revision : 01
430 [009h 0009   1]                     Checksum : 14
431 [00Ah 0010   6]                       Oem ID : " Intel"
432 [010h 0016   8]                 Oem Table ID : "Template"
433 [018h 0024   4]                 Oem Revision : 00000001
434 [01Ch 0028   4]              Asl Compiler ID : "INTL"
435 [020h 0032   4]        Asl Compiler Revision : 20110316


438 [024h 0036  12]      Command/Status Register : <Generic Address Structure>
439 [024h 0036   1]                   Space ID : 01 (SystemIO)
440 [025h 0037   1]                  Bit Width : 08
441 [026h 0038   1]                 Bit Offset : 00
442 [027h 0039   1]       Encoded Access Width : 00 (Undefined/Legacy)
443 [028h 0040   8]                    Address : 0000000000000066

445 [030h 0048  12]               Data Register : <Generic Address Structure>
446 [030h 0048   1]                   Space ID : 01 (SystemIO)
447 [031h 0049   1]                  Bit Width : 08
448 [032h 0050   1]                 Bit Offset : 00
449 [033h 0051   1]       Encoded Access Width : 00 (Undefined/Legacy)
450 [034h 0052   8]                    Address : 0000000000000062

452 [03Ch 0060   4]                        UID : 00000000
453 [040h 0064   1]                  GPE Number : 09
454 [041h 0065  13]                    Namepath : "\_SB.PCI0.EC"
```



```
457 /* Test multiple UEFI support */

459 [0004]                          Signature : "UEFI"    [UEFI Boot Optimization Ta
460 [0004]                       Table Length : 00000036
461 [0001]                           Revision : 01
462 [0001]                           Checksum : 9B
463 [0006]                             Oem ID : " Intel"
464 [0008]                       Oem Table ID : "   UEFI1"
465 [0004]                       Oem Revision : 00000001
466 [0004]                    Asl Compiler ID : "INTL"
467 [0004]              Asl Compiler Revision : 20100528

469 [0016]                    UUID Identifier : 09080706-0504-0706-0809-0A0B0C0D0E0F
470 [0002]                        Data Offset : 0000


473 [004]                          Signature : "UEFI"    /* UEFI Boot Optimization T
474 [004]                       Table Length : 00000036
475 [001]                           Revision : 01
476 [001]                           Checksum : 9B
477 [006]                             Oem ID : " Intel"
478 [008]                       Oem Table ID : "   UEFI2"
479 [004]                       Oem Revision : 00000001
480 [004]                    Asl Compiler ID : "INTL"
481 [004]              Asl Compiler Revision : 20100528

483 [016]                    UUID Identifier : 01234567-0504-0706-0809-0A0B0C0D0E0F
484 [002]                        Data Offset : 0000

486                                    Label : StartRecord
487                                    UINT8 : 4
488                                   UINT16 : $EndRecord - $StartRecord  /* Should
489                                   UINT24 : 123456 | F00000
490                                   UINT32 : 01020304
491                                   UINT56 : 11223344556677
492                                   UINT64 : 0102030405060708 * 4 - 200 / 100 | F0
493                                    Label : EndRecord

495                                    UINT8 : AB
496                                   String : "This is a string"
497                                 DevicePath : "\PCI0\ABCD"
498                                   UINT16 : $StartRecord
499                                  Unicode : "Unicode String"
500                                   UINT64 : $EndRecord * 128

502                                   Buffer : 41 42 43 44 45
503                                   String : ""
504                                     GUID : 03020100-0504-0706-0809-0A0B0C0D0E0F
505 #endif

507 #endif /* __AETABLES_H__ */
```

```
**********************************************************
    4277 Thu Dec 26 13:50:09 2013
new/usr/src/common/acpica/tools/acpihelp/acpihelp.h
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acpihelp.h - Include file for AcpiHelp utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef __ACPIHELP_H
#define __ACPIHELP_H


#include "acpi.h"
#include "accommon.h"
#include "acapps.h"

#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#ifdef WIN32
#include <io.h>
#include <direct.h>
#endif
```



```
#include <errno.h>


#define     AH_DECODE_DEFAULT           0
#define     AH_DECODE_ASL               1
#define     AH_DECODE_ASL_KEYWORD       2
#define     AH_DECODE_PREDEFINED_NAME   3
#define     AH_DECODE_AML               4
#define     AH_DECODE_AML_OPCODE        5
#define     AH_DISPLAY_DEVICE_IDS       6
#define     AH_DECODE_EXCEPTION         7

#define     AH_MAX_ASL_LINE_LENGTH      70
#define     AH_MAX_AML_LINE_LENGTH      100


typedef struct ah_aml_opcode
{
    UINT16          OpcodeRangeStart;
    UINT16          OpcodeRangeEnd;
    char            *OpcodeString;
    char            *OpcodeName;
    char            *Type;
    char            *FixedArguments;
    char            *VariableArguments;
    char            *Grammar;

} AH_AML_OPCODE;

typedef struct ah_asl_operator
{
    char            *Name;
    char            *Syntax;
    char            *Description;

} AH_ASL_OPERATOR;

typedef struct ah_asl_keyword
{
    char            *Name;
    char            *Description;
    char            *KeywordList;

} AH_ASL_KEYWORD;

typedef struct ah_device_id
{
    char            *Name;
    char            *Description;

} AH_DEVICE_ID;


extern const AH_AML_OPCODE          AmlOpcodeInfo[];
extern const AH_ASL_OPERATOR        AslOperatorInfo[];
extern const AH_ASL_KEYWORD         AslKeywordInfo[];
extern BOOLEAN                      AhDisplayAll;

void
AhStrupr (
    char                    *SrcString);

void
AhFindAmlOpcode (
    char                    *Name);
```

```
128 void
129 AhDecodeAmlOpcode (
130     char                    *Name);

132 void
133 AhDecodeException (
134     char                    *Name);

136 void
137 AhFindPredefinedNames (
138     char                    *Name);

140 void
141 AhFindAslOperators (
142     char                    *Name);

144 void
145 AhFindAslKeywords (
146     char                    *Name);

148 void
149 AhDisplayDeviceIds (
150     void);

152 #endif /* __ACPIHELP_H */
```

```
**********************************************************
   34670 Thu Dec 26 13:50:10 2013
new/usr/src/common/acpica/tools/acpihelp/ahamlops.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ahamlops - Table of all known AML opcodes
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpihelp.h"

/*
 * AML opcodes with related syntax and grammar information.
 * This table was extracted from the ACPI specification.
 */
const AH_AML_OPCODE         AmlOpcodeInfo[] =
{
    {0x00,   0x00,   "0x00",           "ZeroOp",             "DataObject",
                                                             NULL},
    {0x01,   0x01,   "0x01",           "OneOp",              "DataObject",
                                                             NULL},
    {0x02,   0x05,   "0x02-0x05",      NULL,                 NULL,
                                                             NULL},
    {0x06,   0x06,   "0x06",           "AliasOp",            "TermObject",
                                                             "DefAlias := Ali
    {0x07,   0x07,   "0x07",           NULL,                 NULL,
                                                             NULL},
```



```
    {0x08,   0x08,   "0x08",           "NameOp",             "TermObject",
                                                             "DefName := Name
    {0x09,   0x09,   "0x09",           NULL,                 NULL,
                                                             NULL},
    {0x0A,   0x0A,   "0x0A",           "BytePrefix",         "DataObject",
                                                             "ByteConst := By
    {0x0B,   0x0B,   "0x0B",           "WordPrefix",         "DataObject",
                                                             "WordConst := Wo
    {0x0C,   0x0C,   "0x0C",           "DWordPrefix",        "DataObject",
                                                             "DWordConst := D
    {0x0D,   0x0D,   "0x0D",           "StringPrefix",       "DataObject",
                                                             "String := Strin
    {0x0E,   0x0E,   "0x0E",           "QWordPrefix",        "DataObject",
                                                             "QWordConst := Q
    {0x0F,   0x0F,   "0x0F",           NULL,                 NULL,
                                                             NULL},
    {0x10,   0x10,   "0x10",           "ScopeOp",            "TermObject",
                                                             "DefScope := Sco
    {0x11,   0x11,   "0x11",           "BufferOp",           "TermObject",
                                                             "DefBuffer := Bu
    {0x12,   0x12,   "0x12",           "PackageOp",          "TermObject",
                                                             "DefPackage := P
    {0x13,   0x13,   "0x13",           "VarPackageOp",       "TermObject",
                                                             "DefVarPackage :
    {0x14,   0x14,   "0x14",           "MethodOp",           "TermObject",
                                                             "DefMethod := Me
    {0x15,   0x2D,   "0x15-0x2D",      NULL,                 NULL,
                                                             NULL},
    {0x2E,   0x2E,   "0x2E",           "DualNamePrefix",     "NameObject",
                                                             "DualNamePath :=
    {0x2F,   0x2F,   "0x2F",           "MultiNamePrefix",    "NameObject",
                                                             "MultiNamePath :
    {0x30,   0x39,   "0x30-0x39",      "DigitChar",          "NameObject",
                                                             NULL},
    {0x3A,   0x40,   "0x3A-0x40",      NULL,                 NULL,
                                                             NULL},
    {0x41,   0x5A,   "0x41-0x5A",      "NameChar",           "NameObject",
                                                             NULL},
    {0x5B,   0x5B,   "0x5B",           "ExtOpPrefix",        "DataObject",
                                                             NULL},
    {0x5B00, 0x5B00, "0x5B00",         NULL,                 NULL,
                                                             NULL},
    {0x5B01, 0x5B01, "0x5B01",         "MutexOp",            "TermObject",
                                                             "DefMutex := Mut
    {0x5B02, 0x5B02, "0x5B02",         "EventOp",            "TermObject",
                                                             "DefEvent := Eve
    {0x5B12, 0x5B12, "0x5B12",         "CondRefOfOp",        "TermObject",
                                                             "DefCondRefOf :=
    {0x5B13, 0x5B13, "0x5B13",         "CreateFieldOp",      "TermObject",
                                                             "DefCreateField
    {0x5B1F, 0x5B1F, "0x5B1F",         "LoadTableOp",        "TermObject",
                                                             "DefLoadTable :=
    {0x5B20, 0x5B20, "0x5B20",         "LoadOp",             "TermObject",
                                                             "DefLoad := Load
    {0x5B21, 0x5B21, "0x5B21",         "StallOp",            "TermObject",
                                                             "DefStall := Sta
    {0x5B22, 0x5B22, "0x5B22",         "SleepOp",            "TermObject",
                                                             "DefSleep := Sle
    {0x5B23, 0x5B23, "0x5B23",         "AcquireOp",          "TermObject",
                                                             "DefAcquire := A
    {0x5B24, 0x5B24, "0x5B24",         "SignalOp",           "TermObject",
                                                             "DefSignal := Si
    {0x5B25, 0x5B25, "0x5B25",         "WaitOp",             "TermObject",
                                                             "DefWait := Wait
    {0x5B26, 0x5B26, "0x5B26",         "ResetOp",            "TermObject",
                                                             "DefReset := Res
```

```
128     {0x5B27, 0x5B27, "0x5B27",          "ReleaseOp",          "TermObject",
129                                                               "DefRelease := R
130     {0x5B28, 0x5B28, "0x5B28",          "FromBCDOp",          "TermObject",
131                                                               "DefFromBCD := F
132     {0x5B29, 0x5B29, "0x5B29",          "ToBCD",              "TermObject",
133                                                               "DefToBCD := ToB
134     {0x5B2A, 0x5B2A, "0x5B2A",          "UnloadOp",           "TermObject",
135                                                               "DefUnload := Un
136     {0x5B30, 0x5B30, "0x5B30",          "RevisionOp",         "DataObject",
137                                                               NULL},
138     {0x5B31, 0x5B31, "0x5B31",          "DebugOp",            "DebugObject",
139                                                               NULL},
140     {0x5B32, 0x5B32, "0x5B32",          "FatalOp",            "TermObject",
141                                                               "DefFatal := Fat
142     {0x5B33, 0x5B33, "0x5B33",          "TimerOp",            "TermObject",
143                                                               "DefTimer := Tim
144     {0x5B80, 0x5B80, "0x5B80",          "OpRegionOp",         "TermObject",
145                                                               "DefOpRegion :=
146     {0x5B81, 0x5B81, "0x5B81",          "FieldOp",            "TermObject",
147                                                               "DefField := Fie
148     {0x5B82, 0x5B82, "0x5B82",          "DeviceOp",           "TermObject",
149                                                               "DefDevice := De
150     {0x5B83, 0x5B83, "0x5B83",          "ProcessorOp",        "TermObject",
151                                                               "DefProcessor :=
152     {0x5B84, 0x5B84, "0x5B84",          "PowerResOp",         "TermObject",
153                                                               "DefPowerRes :=
154     {0x5B85, 0x5B85, "0x5B85",          "ThermalZoneOp",      "TermObject",
155                                                               "DefThermalZone
156     {0x5B86, 0x5B86, "0x5B86",          "IndexFieldOp",       "TermObject",
157                                                               "DefIndexField :
158     {0x5B87, 0x5B87, "0x5B87",          "BankFieldOp",        "TermObject",
159                                                               "DefBankField :=
160     {0x5B88, 0x5B88, "0x5B88",          "DataRegionOp",       "TermObject",
161                                                               "DefDataRegion :
162     {0x5B89, 0x5BFF, "0x5B89-0x5BFF",   NULL,                 NULL,
163                                                               NULL},
164     {0x5C,   0x5C,   "0x5C",            "RootChar",           "NameObject",
165                                                               NULL},
166     {0x5D,   0x5D,   "0x5D",            NULL,                 NULL,
167                                                               NULL},
168     {0x5E,   0x5E,   "0x5E",            "ParentPrefixChar",   "NameObject",
169                                                               NULL},
170     {0x5F,   0x5F,   "0x5F",            "NameChar",           "NameObject",
171                                                               NULL},
172     {0x60,   0x60,   "0x60",            "Local0Op",           "LocalObject",
173                                                               NULL},
174     {0x61,   0x61,   "0x61",            "Local1Op",           "LocalObject",
175                                                               NULL},
176     {0x62,   0x62,   "0x62",            "Local2Op",           "LocalObject",
177                                                               NULL},
178     {0x63,   0x63,   "0x63",            "Local3Op",           "LocalObject",
179                                                               NULL},
180     {0x64,   0x64,   "0x64",            "Local4Op",           "LocalObject",
181                                                               NULL},
182     {0x65,   0x65,   "0x65",            "Local5Op",           "LocalObject",
183                                                               NULL},
184     {0x66,   0x66,   "0x66",            "Local6Op",           "LocalObject",
185                                                               NULL},
186     {0x67,   0x67,   "0x67",            "Local7Op",           "LocalObject",
187                                                               NULL},
188     {0x68,   0x68,   "0x68",            "Arg0Op",             "ArgObject",
189                                                               NULL},
190     {0x69,   0x69,   "0x69",            "Arg1Op",             "ArgObject",
191                                                               NULL},
192     {0x6A,   0x6A,   "0x6A",            "Arg2Op",             "ArgObject",
193                                                               NULL},
```



```
194     {0x6B,   0x6B,   "0x6B",            "Arg3Op",             "ArgObject",
195                                                               NULL},
196     {0x6C,   0x6C,   "0x6C",            "Arg4Op",             "ArgObject",
197                                                               NULL},
198     {0x6D,   0x6D,   "0x6D",            "Arg5Op",             "ArgObject",
199                                                               NULL},
200     {0x6E,   0x6E,   "0x6E",            "Arg6Op",             "ArgObject",
201                                                               NULL},
202     {0x6F,   0x6F,   "0x6F",            NULL,                 NULL,
203                                                               NULL},
204     {0x70,   0x70,   "0x70",            "StoreOp",            "TermObject",
205                                                               "DefStore := Sto
206     {0x71,   0x71,   "0x71",            "RefOfOp",            "TermObject",
207                                                               "DefRefOf := Ref
208     {0x72,   0x72,   "0x72",            "AddOp",              "TermObject",
209                                                               "DefAdd := AddOp
210     {0x73,   0x73,   "0x73",            "ConcatOp",           "TermObject",
211                                                               "DefConcat := Co
212     {0x74,   0x74,   "0x74",            "SubtractOp",         "TermObject",
213                                                               "DefSubtract :=
214     {0x75,   0x75,   "0x75",            "IncrementOp",        "TermObject",
215                                                               "DefIncrement :=
216     {0x76,   0x76,   "0x76",            "DecrementOp",        "TermObject",
217                                                               "DefDecrement :=
218     {0x77,   0x77,   "0x77",            "MultiplyOp",         "TermObject",
219                                                               "DefMultiply :=
220     {0x78,   0x78,   "0x78",            "DivideOp",           "TermObject",
221                                                               "DefDivide := Di
222     {0x79,   0x79,   "0x79",            "ShiftLeftOp",        "TermObject",
223                                                               "DefShiftLeft :=
224     {0x7A,   0x7A,   "0x7A",            "ShiftRightOp",       "TermObject",
225                                                               "DefShiftRight :
226     {0x7B,   0x7B,   "0x7B",            "AndOp",              "TermObject",
227                                                               "DefAnd := AndOp
228     {0x7C,   0x7C,   "0x7C",            "NandOp",             "TermObject",
229                                                               "DefNAnd := Nand
230     {0x7D,   0x7D,   "0x7D",            "OrOp",               "TermObject",
231                                                               "DefOr := OrOp O
232     {0x7E,   0x7E,   "0x7E",            "NorOp",              "TermObject",
233                                                               "DefNOr := NorOp
234     {0x7F,   0x7F,   "0x7F",            "XorOp",              "TermObject",
235                                                               "DefXOr := XorOp
236     {0x80,   0x80,   "0x80",            "NotOp",              "TermObject",
237                                                               "DefNot := NotOp
238     {0x81,   0x81,   "0x81",            "FindSetLeftBitOp",   "TermObject",
239                                                               "DefFindSetLeftB
240     {0x82,   0x82,   "0x82",            "FindSetRightBitOp",  "TermObject",
241                                                               "DefFindSetRight
242     {0x83,   0x83,   "0x83",            "DerefOfOp",          "TermObject",
243                                                               "DefDerefOf := D
244     {0x84,   0x84,   "0x84",            "ConcatResOp",        "TermObject",
245                                                               "DefConcatRes :=
246     {0x85,   0x85,   "0x85",            "ModOp",              "TermObject",
247                                                               "DefMod := ModOp
248     {0x86,   0x86,   "0x86",            "NotifyOp",           "TermObject",
249                                                               "DefNotify := No
250     {0x87,   0x87,   "0x87",            "SizeOfOp",           "TermObject",
251                                                               "DefSizeOf := Si
252     {0x88,   0x88,   "0x88",            "IndexOp",            "TermObject",
253                                                               "DefIndex := Ind
254     {0x89,   0x89,   "0x89",            "MatchOp",            "TermObject",
255                                                               "DefMatch := Mat
256     {0x8A,   0x8A,   "0x8A",            "CreateDWordFieldOp", "TermObject",
257                                                               "DefCreateDWordF
258     {0x8B,   0x8B,   "0x8B",            "CreateWordFieldOp",  "TermObject",
259                                                               "DefCreateWordFi
```

```
260     {0x8C,   0x8C,   "0x8C",            "CreateByteFieldOp",    "TermObject",
261                                                                 "DefCreateByteFi
262     {0x8D,   0x8D,   "0x8D",            "CreateBitFieldOp",     "TermObject",
263                                                                 "DefCreateBitFie
264     {0x8E,   0x8E,   "0x8E",            "ObjectTypeOp",         "TermObject",
265                                                                 "DefObjectType :
266     {0x8F,   0x8F,   "0x8F",            "CreateQWordFieldOp",   "TermObject",
267                                                                 "DefCreateQWordF
268     {0x90,   0x90,   "0x90",            "LAndOp",               "TermObject",
269                                                                 "DefLAnd := Land
270     {0x91,   0x91,   "0x91",            "LOrOp",                "TermObject",
271                                                                 "DefLOr := LorOp
272     {0x92,   0x92,   "0x92",            "LNotOp",               "TermObject",
273                                                                 "DefLNot := Lnot
274     {0x9293, 0x9293, "0x9293",          "LNotEqualOp",          "TermObject",
275                                                                 "DefLNotEqual :=
276     {0x9294, 0x9294, "0x9294",          "LLessEqualOp",         "TermObject",
277                                                                 "DefLLessEqual :
278     {0x9295, 0x9295, "0x9295",          "LGreaterEqualOp",      "TermObject",
279                                                                 "DefLGreaterEqua
280     {0x93,   0x93,   "0x93",            "LEqualOp",             "TermObject",
281                                                                 "DefLEqual := Le
282     {0x94,   0x94,   "0x94",            "LGreaterOp",           "TermObject",
283                                                                 "DefLGreater :=
284     {0x95,   0x95,   "0x95",            "LLessOp",              "TermObject",
285                                                                 "DefLLess := Lle
286     {0x96,   0x96,   "0x96",            "ToBufferOp",           "TermObject",
287                                                                 "DefToBuffer :=
288     {0x97,   0x97,   "0x97",            "ToDecimalStringOp",    "TermObject",
289                                                                 "DefToDecimalStr
290     {0x98,   0x98,   "0x98",            "ToHexStringOp",        "TermObject",
291                                                                 "DefToHexString
292     {0x99,   0x99,   "0x99",            "ToIntegerOp",          "TermObject",
293                                                                 "DefToInteger :=
294     {0x9A,   0x9B,   "0x9A-0x9B",       NULL,                   NULL,
295                                                                 NULL},
296     {0x9C,   0x9C,   "0x9C",            "ToStringOp",           "TermObject",
297                                                                 "DefToString :=
298     {0x9D,   0x9D,   "0x9D",            "CopyObjectOp",         "TermObject",
299                                                                 "DefCopyObject :
300     {0x9E,   0x9E,   "0x9E",            "MidOp",                "TermObject",
301                                                                 "DefMid := MidOp
302     {0x9F,   0x9F,   "0x9F",            "ContinueOp",           "TermObject",
303                                                                 "DefContinue :=
304     {0xA0,   0xA0,   "0xA0",            "IfOp",                 "TermObject",
305                                                                 "DefIfElse := If
306     {0xA1,   0xA1,   "0xA1",            "ElseOp",               "TermObject",
307                                                                 "DefElse := Noth
308     {0xA2,   0xA2,   "0xA2",            "WhileOp",              "TermObject",
309                                                                 "DefWhile := Whi
310     {0xA3,   0xA3,   "0xA3",            "NoopOp",               "TermObject",
311                                                                 "DefNoop := Noop
312     {0xA4,   0xA4,   "0xA4",            "ReturnOp",             "TermObject",
313                                                                 "DefReturn := Re
314     {0xA5,   0xA5,   "0xA5",            "BreakOp",              "TermObject",
315                                                                 "DefBreak := Bre
316     {0xA6,   0xCB,   "0xA6-0xCB",       NULL,                   NULL,
317                                                                 NULL},
318     {0xCC,   0xCC,   "0xCC",            "BreakPointOp",         "TermObject",
319                                                                 "DefBreakPoint :
320     {0xCD,   0xFE,   "0xCD-0xFE",       NULL,                   NULL,
321                                                                 NULL},
322     {0xFF,   0xFF,   "0xFF",            "OnesOp",               "DataObject",
323                                                                 NULL},
324     {0, 0, NULL,  NULL, NULL, NULL, NULL, NULL}
325 };
```

```
**********************************************************
    7842 Thu Dec 26 13:50:10 2013
new/usr/src/common/acpica/tools/acpihelp/ahaslkey.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ahaslkey - Table of all known ASL non-operator keywords
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpihelp.h"

/*
 * ASL Keyword types and associated actual keywords.
 * This table was extracted from the ACPI specification.
 */
const AH_ASL_KEYWORD        AslKeywordInfo[] =
{
    {"AccessAttribKeyword", "Serial Bus Attributes (with legacy SMBus aliases)",
        ":= AttribQuick (SMBusQuick) | AttribSendReceive (SMBusSendReceive) | "
        "AttribByte (SMBusByte) | AttribWord (SMBusWord) | "
        "AttribBlock (SMBusBlock) | AttribProcessCall (SMBusProcessCall) | "
        "AttribBlockProcessCall (SMBusProcessCall)"},
    {"AccessTypeKeyword", "Field Access Types",
        ":= AnyAcc | ByteAcc | WordAcc | DWordAcc | QWordAcc | BufferAcc"},
    {"AddressingModeKeyword", "Mode - Resource Descriptors",
        ":= AddressingMode7Bit | AddressingMode10Bit"},
    {"AddressKeyword", "ACPI memory range types",
```



```
        ":= AddressRangeMemory | AddressRangeReserved | "
        "AddressRangeNVS | AddressRangeACPI"},
    {"AddressSpaceKeyword", "Operation Region Address Space Types",
        ":= RegionSpaceKeyword | FFixedHW"},
    {"BusMasterKeyword", "DMA Bus Mastering",
        ":= BusMaster | NotBusMaster"},
    {"ByteLengthKeyword", "Bits per Byte - Resource Descriptors",
        ":= DataBitsFive | DataBitsSix | DataBitsSeven | DataBitsEight | DataBit
    {"ClockPhaseKeyword", "Resource Descriptors",
        ":= ClockPhaseFirst | ClockPhaseSecond"},
    {"ClockPolarityKeyword", "Resource Descriptors",
        ":= ClockPolarityLow | ClockPolarityHigh"},
    {"DecodeKeyword", "Type of Memory Decoding - Resource Descriptors",
        ":= SubDecode | PosDecode"},
    {"DmaTypeKeyword", "DMA Types - DMA Resource Descriptor",
        ":= Compatibility | TypeA | TypeB | TypeF"},
    {"EndianKeyword", "Endian type - Resource Descriptor",
        ":= BigEndian | LittleEndian"},
    {"ExtendedAttribKeyword", "Extended Bus Attributes",
        ":= AttribBytes (AccessLength) | AttribRawBytes (AccessLength) | "
        "AttribRawProcessBytes (AccessLength)"},
    {"FlowControlKeyword", "Resource Descriptor",
        ":= FlowControlNone | FlowControlXon | FlowControlHardware"},
    {"InterruptLevelKeyword", "Interrupt Active Types",
        ":= ActiveHigh | ActiveLow | ActiveBoth"},
    {"InterruptTypeKeyword", "Interrupt Types",
        ":= Edge | Level"},
    {"IoDecodeKeyword", "I/O Decoding - IO Resource Descriptor",
        ":= Decode16 | Decode10"},
    {"IoRestrictionKeyword", "I/O Restriction - GPIO Resource Descriptors",
        ":= IoRestrictionNone | IoRestrictionInputOnly | "
        "IoRestrictionOutputOnly | IoRestrictionNoneAndPreserve"},
    {"LockRuleKeyword", "Global Lock use for Field Operator",
        ":= Lock | NoLock"},
    {"MatchOpKeyword", "Types for Match Operator",
        ":= MTR | MEQ | MLE | MLT | MGE | MGT"},
    {"MaxKeyword", "Max Range Type - Resource Descriptors",
        ":= MaxFixed | MaxNotFixed"},
    {"MemTypeKeyword", "Memory Types - Resource Descriptors",
        ":= Cacheable | WriteCombining | Prefetchable | NonCacheable"},
    {"MinKeyword", "Min Range Type - Resource Descriptors",
        ":= MinFixed | MinNotFixed"},
    {"ObjectTypeKeyword", "ACPI Object Types",
        ":= UnknownObj | IntObj | StrObj | BuffObj | PkgObj | FieldUnitObj | "
        "DeviceObj | EventObj | MethodObj | MutexObj | OpRegionObj | PowerResObj
        "ProcessorObj | ThermalZoneObj | BuffFieldObj | DDBHandleObj"},
    {"ParityKeyword", "Resource Descriptors",
        ":= ParityTypeNone | ParityTypeSpace | ParityTypeMark | "
        "ParityTypeOdd | ParityTypeEven"},
    {"PinConfigKeyword", "Pin Configuration - GPIO Resource Descriptors",
        ":= PullDefault | PullUp | PullDown | PullNone"},
    {"PolarityKeyword", "Resource Descriptors",
        ":= PolarityHigh | PolarityLow"},
    {"RangeTypeKeyword", "I/O Range Types - Resource Descriptors",
        ":= ISAOnlyRanges | NonISAOnlyRanges | EntireRange"},
    {"ReadWriteKeyword", "Memory Access Types - Resource Descriptors",
        ":= ReadWrite | ReadOnly"},
    {"RegionSpaceKeyword", "Operation Region Address Space Types",
        ":= UserDefRegionSpace | SystemIO | SystemMemory | PCI_Config | "
        "EmbeddedControl | SMBus | SystemCMOS | PciBarTarget | IPMI | "
        "GeneralPurposeIo, GenericSerialBus"},
    {"ResourceTypeKeyword", "Resource Usage - Resource Descriptors",
        ":= ResourceConsumer | ResourceProducer"},
    {"SerializeRuleKeyword", "Control Method Serialization",
        ":= Serialized | NotSerialized"},
    {"ShareTypeKeyword", "Interrupt Sharing - Resource Descriptors",
```

```
128         ":= Shared | Exclusive | SharedAndWake | ExclusiveAndWake"},
129     {"SlaveModeKeyword", "Resource Descriptors",
130         ":= ControllerInitiated | DeviceInitiated"},
131     {"StopBitsKeyword", "Resource Descriptors",
132         ":= StopBitsZero | StopBitsOne | StopBitsOnePlusHalf | StopBitsTwo"},
133     {"TransferWidthKeyword", "DMA Widths - Fixed DMA Resource Descriptor",
134         ":= Width8bit | Width16bit | Width32bit | Width64bit | "
135         "Width128bit | Width256bit"},
136     {"TranslationKeyword", "Translation Density Types - Resource Descriptors",
137         ":= SparseTranslation | DenseTranslation"},
138     {"TypeKeyword", "Translation Types - Resource Descriptors",
139         ":= TypeTranslation | TypeStatic"},
140     {"UpdateRuleKeyword", "Field Update Rules",
141         ":= Preserve | WriteAsOnes | WriteAsZeros"},
142     {"UserDefRegionSpace", "User defined address spaces",
143         ":= IntegerData => 0x80 - 0xFF"},
144     {"WireModeKeyword", "SPI Wire Mode - Resource Descriptors",
145         ":= ThreeWireMode | FourWireMode"},
146     {"XferTypeKeyword", "DMA Transfer Types",
147         ":= Transfer8 | Transfer16 | Transfer8_16"},
148     {NULL, NULL, NULL}
149 };
```

```
**********************************************************
   28994 Thu Dec 26 13:50:10 2013
new/usr/src/common/acpica/tools/acpihelp/ahaslops.c
acpica-unix2-20130823
**********************************************************
/******************************************************************************
 *
 * Module Name: ahaslops - Table of all known ASL operators
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpihelp.h"

/*
 * ASL operators with syntax (directly from ACPI specification).
 * Note: All tokens require a space separator.
 * Long lines are automatically split during output.
 */
const AH_ASL_OPERATOR       AslOperatorInfo[] =
{
    {"AccessAs",                "(AccessType, AccessAttribKeyword | "
                                "ExtendedAttribKeyword (AccessLength))",
                                "ChangeFieldUnitAccess"},
    {"Acquire",                 "(SyncObject, TimeoutValue) => Boolean",
                                "Acquire a mutex"},
    {"Add",                     "(Addend1, Addend2, Result) => Integer",
                                "Integer Add"},
    {"Alias",                   "(SourceObject, AliasObject)",
                                "Define a name alias"},
```



```
    {"And",                     "(Source1, Source2, Result) => Integer",
                                "Integer Bitwise And"},
    {"Arg",                     "Arg0 - Arg6",
                                "Method argument data objects"},
    {"BankField",               "(RegionName, BankName, BankValue, "
                                "AccessTypeKeyword, LockRuleKeyword, "
                                "UpdateRuleKeyword) {FieldUnitList}",
                                "Declare fields in a banked configuration object
    {"Break",                   "No parameters",
                                "Continue following the innermost enclosing Whil
    {"BreakPoint",              "No parameters",
                                "Used for debugging, stops execution in the debu
    {"Buffer",                  "(BufferSize) {String or ByteList} => Buffer",
                                "Declare Buffer object"},
    {"Case",                    "(Value) {TermList}",
                                "Expression for conditional execution"},
    {"Concatenate",             "(Source1, Source2, Result) => ComputationalData
                                "Concatenate two strings, integers or buffers"},
    {"ConcatenateResTemplate",  "(Source1, Source2, Result) => Buffer",
                                "Concatenate two resource templates"},
    {"CondRefOf",               "(Source, Result) => Boolean",
                                "Conditional reference to an object"},
    {"Connection",              "(ResourceMacro)",
                                "Associate connection with FieldUnits within a F
    {"Continue",                "No parameters",
                                "Continue innermost enclosing While loop"},
    {"CopyObject",              "(Source, Destination) => DataRefObject",
                                "Copy and existing object"},
    {"CreateBitField",          "(SourceBuffer, BitIndex, BitFieldName)",
                                "Declare a bit field object of a buffer object"}
    {"CreateByteField",         "(SourceBuffer, ByteIndex, ByteFieldName)",
                                "Declare a byte field object of a buffer object"
    {"CreateDWordField",        "(SourceBuffer, ByteIndex, DWordFieldName)",
                                "Declare a DWord field object of a buffer object
    {"CreateField",             "(SourceBuffer, BitIndex, NumBits, FieldName)",
                                "Declare an arbitrary length bit field of a buff
    {"CreateQWordField",        "(SourceBuffer, ByteIndex, QWordFieldName)",
                                "Declare a QWord field object of a buffer object
    {"CreateWordField",         "(SourceBuffer, ByteIndex, WordFieldName)",
                                "Declare a Word field object of a buffer object"
    {"DataTableRegion",         "(RegionName, SignatureString, OemIDString, OemT
                                "Declare a Data Table Region"},
    {"Debug",                   "No parameters",
                                "Debugger output"},
    {"Decrement",               "(Minuend) => Integer",
                                "Decrement an Integer"},
    {"Default",                 "{TermList}",
                                "Default execution path in Switch()"},
    {"DefinitionBlock",         "(AmlFileName, TableSignature, ComplianceRevisio
                                "OemId, TableId, OemRevision) {TermList}",
                                "Declare a Definition Block"},
    {"DerefOf",                 "(Source) => Object",
                                "Dereference an object reference"},
    {"Device",                  "(DeviceName) {ObjectList}",
                                "Declare a bus/device object"},
    {"Divide",                  "(Dividend, Divisor, Remainder, Result) => Integ
                                "Integer Divide"},
    {"DMA",                     "(DmaTypeKeyword, BusMasterKeyword, XferTypeKeyw
                                "DescriptorName) {DmaChannelList} => Buffer",
                                "DMA Resource Descriptor macro"},
    {"DWordIO",                 "(ResourceTypeKeyword, MinKeyword, MaxKeyword, "
                                "DecodeKeyword, RangeTypeKeyword, AddressGranula
                                "AddressMinimum, AddressMaximum, AddressTranslat
                                "RangeLength, ResourceSourceIndex, "
                                "ResourceSource, DescriptorName, TypeKeyword, Tr
                                "DWord IO Resource Descriptor macro"},
```

```
    {"DWordMemory",             "(ResourceTypeKeyword, DecodeKeyword, MinKeyword
                                "MaxKeyword, MemTypeKeyword, ReadWriteKeyword, "
                                "AddressGranularity, AddressMinimum, AddressMaxi
                                "RangeLength, ResourceSourceIndex, ResourceSourc
                                "TypeKeyword)",
                                "DWord Memory Resource Descriptor macro"},
    {"DWordSpace",              "(ResourceType, ResourceTypeKeyword, DecodeKeywo
                                "MinKeyword, MaxKeyword, TypeSpecificFlags, "
                                "AddressGranularity, AddressMinimum, AddressMaxi
                                "AddressTranslation, RangeLength, "
                                "ResourceSourceIndex, ResourceSource, Descriptor
                                "DWord Space Resource Descriptor macro"},
    {"EISAID",                  "(EisaIdString) => DWordConst",
                                "EISA ID String to Integer conversion macro"},
    {"Else",                    "{TermList}",
                                "Alternate conditional execution"},
    {"ElseIf",                  "(Predicate)",
                                "Conditional execution"},
    {"EndDependentFn",          "() => Buffer",
                                "End Dependent Function Resource Descriptor macr
    {"Event",                   "(EventName)",
                                "Declare an event synchronization object"},
    {"ExtendedIO",              "(ResourceTypeKeyword, MinKeyword, MaxKeyword, "
                                "DecodeKeyword, RangeTypeKeyword, AddressGranula
                                "AddressMinimum, AddressMaximum, AddressTranslat
                                "TypeSpecificAttributes, DescriptorName, TypeKey
                                "Extended IO Resource Descriptor macro"},
    {"ExtendedMemory",          "(ResourceTypeKeyword, DecodeKeyword, MinKeyword
                                "MaxKeyword, MemTypeKeyword, ReadWriteKeyword, "
                                "AddressGranularity, AddressMinimum, AddressMaxi
                                "RangeLength, TypeSpecificAttributes, Descriptor
                                "AddressKeyword, TypeKeyword)",
                                "Extended Memory Resource Descriptor macro"},
    {"ExtendedSpace",           "(ResourceType, ResourceTypeKeyword, DecodeKeywo
                                "MinKeyword, MaxKeyword, TypeSpecificFlags, "
                                "AddressGranularity, AddressMinimum, AddressMaxi
                                "RangeLength, TypeSpecificAttributes, Descriptor
                                "Extended Space Resource Descriptor macro"},
    {"External",                "(ObjectName, ObjectTypeKeyword, ReturnType, Par
                                "Declare external objects"},
    {"Fatal",                   "(Type, Code, Arg)",
                                "Fatal error check"},
    {"Field",                   "(RegionName, AccessTypeKeyword, LockRuleKeyword
                                "UpdateRuleKeyword) {FieldUnitList}",
                                "Declare fields of an operation region object"},
    {"FindSetLeftBit",          "(Source, Result) => Integer",
                                "Index of first least significant bit set"},
    {"FindSetRightBit",         "(Source, Result) => Integer",
                                "Index of first most significant bit set"},
    {"FixedDMA",                "(DmaRequestLine, Channel, TransferWidthKeyword,
                                "Fixed DMA Resource Descriptor macro"},
    {"FixedIO",                 "(AddressBase, RangeLength, DescriptorName) => B
                                "Fixed I/O Resource Descriptor macro"},
    {"FromBCD",                 "(BCDValue, Result) => Integer",
                                "Convert from BCD to numeric"},
    {"Function",                "(FunctionName, ReturnType, ParameterTypes) {Ter
                                "Declare control method"},
    {"GpioInt",                 "(InterruptTypeKeyword, InterruptLevelKeyword, "
                                "ShareTypeKeyword, PinConfigKeyword, "
                                "DebounceTimeout, ResourceSource, "
                                "ResourceSourceIndex, ResourceTypeKeyword, Descr
                                "RawDataBuffer() {VendorData}) {Pin}",
                                "GPIO Interrupt Connection Resource Descriptor M
    {"GpioIo",                  "(ShareTypeKeyword, PinConfigKeyword, DebounceTi
                                "IoRestrictionKeyword, ResourceSource, "
                                "ResourceSourceIndex, ResourceTypeKeyword, Descr
```



```
                                "RawDataBuffer() {VendorData}) {PinList}",
                                "GPIO I/O Connection Resource Descriptor Macro"}
    {"I2cSerialBus",            "(SlaveAddress, SlaveModeKeyword, ConnectionSpee
                                "AddressingModeKeyword, ResourceSource, "
                                "ResourceSourceIndex, ResourceTypeKeyword, Descr
                                "RawDataBuffer() {VendorData})",
                                "I2C Serial Bus Connection Resource Descriptor M
    {"If",                      "(Predicate) {TermList}",
                                "Conditional execution"},
    {"Include",                 "(FilePathName)",
                                "Include another ASL file"},
    {"Increment",               "(Addend) => Integer",
                                "Increment a Integer"},
    {"Index",                   "(Source, Index, Destination) => ObjectReference
                                "Indexed Reference to member object"},
    {"IndexField",              "(IndexName, DataName, AccessTypeKeyword, LockRu
                                "UpdateRuleKeyword) {FieldUnitList}",
                                "Declare Index/Data Fields"},
    {"Interrupt",               "(ResourceTypeKeyword, InterruptTypeKeyword, Int
                                "ShareTypeKeyword, ResourceSourceIndex, "
                                "ResourceSource, DescriptorName) {InterruptList}
                                "Interrupt Resource Descriptor macro"},
    {"IO",                      "(IoDecodeKeyword, AddressMin, AddressMax, Addre
                                "RangeLength, DescriptorName) => Buffer",
                                "IO Resource Descriptor macro"},
    {"IRQ",                     "(InterruptTypeKeyword, InterruptLevelKeyword, S
                                "DescriptorName) {InterruptList} => Buffer",
                                "Interrupt Resource Descriptor macro"},
    {"IRQNoFlags",              "(DescriptorName) {InterruptList} => Buffer",
                                "Short Interrupt Resource Descriptor macro"},
    {"LAnd",                    "(Source1, Source2) => Boolean",
                                "Logical And"},
    {"LEqual",                  "(Source1, Source2) => Boolean",
                                "Logical Equal"},
    {"LGreater",                "(Source1, Source2) => Boolean",
                                "Logical Greater"},
    {"LGreaterEqual",           "(Source1, Source2) => Boolean",
                                "Logical Not less"},
    {"LLess",                   "(Source1, Source2) => Boolean",
                                "Logical Less"},
    {"LLessEqual",              "(Source1, Source2) => Boolean",
                                "Logical Not greater"},
    {"LNot",                    "(Source) => Boolean",
                                "Logical Not"},
    {"LNotEqual",               "(Source1, Source2) => Boolean",
                                "Logical Not equal"},
    {"Load",                    "(Object, DDBHandle)",
                                "Load differentiating definition block"},
    {"LoadTable",               "(SignatureString, OemIdString, OemTableIdString
                                "ParameterPathString, ParameterData) => DDBHandl
                                "Load Table from RSDT/XSDT"},
    {"Local",                   "Local0 - Local7",
                                "Method local data objects"},
    {"LOr",                     "(Source1, Source2) => Boolean",
                                "Logical Or"},
    {"Match",                   "(SearchPackage, MatchOpKeyword, MatchObject1, M
                                "MatchObject2, StartIndex) => Ones | Integer",
                                "Search for match in package array"},
    {"Memory24",                "(ReadWriteKeyword, AddressMinimum, AddressMaxim
                                "RangeLength, DescriptorName)",
                                "Memory Resource Descriptor macro"},
    {"Memory32",                "(ReadWriteKeyword, AddressMinimum, AddressMaxim
                                "RangeLength, DescriptorName)",
                                "Memory Resource Descriptor macro"},
    {"Memory32Fixed",           "(ReadWriteKeyword, AddressBase, RangeLength, De
                                "Memory Resource Descriptor macro"},
```

```
260     {"Method",                  "(MethodName, NumArgs, SerializeRuleKeyword, "
261                                 "SyncLevel, ReturnType, ParameterTypes) "
262                                 "{TermList}",
263                                 "Declare a control method"},
264     {"Mid",                     "(Source, Index, Length, Result) => Buffer or St
265                                 "Return a portion of buffer or string"},
266     {"Mod",                     "(Dividend, Divisor, Result) => Integer",
267                                 "Integer Modulo"},
268     {"Multiply",                "(Multiplicand, Multiplier, Result) => Integer",
269                                 "Integer Multiply"},
270     {"Mutex",                   "(MutexName, SyncLevel)",
271                                 "Declare a mutex synchronization object"},
272     {"Name",                    "(ObjectName, Object)",
273                                 "Declare a Named object"},
274     {"NAnd",                    "(Source1, Source2, Result) => Integer",
275                                 "Integer Bitwise Nand"},
276     {"NoOp",                    "No parameters",
277                                 "No operation"},
278     {"NOr",                     "(Source1, Source2, Result) => Integer",
279                                 "Integer Bitwise Nor"},
280     {"Not",                     "(Source, Result) => Integer",
281                                 "Integer Bitwise Not"},
282     {"Notify",                  "(Object, NotificationValue)",
283                                 "Notify Object of event"},
284     {"ObjectType",              "(Object) => Integer",
285                                 "Type of object"},
286     {"Offset",                  "(ByteOffset)",
287                                 "Change Current Field Unit Offset"},
288     {"One",                     "=> Integer",
289                                 "Constant One Object (1)"},
290     {"Ones",                    "=> Integer",
291                                 "Constant Ones Object (0xFFFFFFFF or 0xFFFFFFFFF
292     {"OperationRegion",         "(RegionName, RegionSpaceKeyword, Offset, Length
293                                 "Declare an operational region"},
294     {"Or",                      "(Source1, Source2, Result) => Integer",
295                                 "Integer Bitwise Or"},
296     {"Package",                 "(NumElements) {PackageList} => Package",
297                                 "Declare a package object"},
298     {"PowerResource",           "(ResourceName, SystemLevel, ResourceOrder) {Obj
299                                 "Declare a power resource object"},
300     {"Processor",               "(ProcessorName, ProcessorID, PBlockAddress, Pbl
301                                 "Declare a processor package"},
302     {"QWordIO",                 "(ResourceTypeKeyword, MinKeyword, MaxKeyword, D
303                                 "RangeTypeKeyword, AddressGranularity, "
304                                 "AddressMinimum, AddressMaximum, AddressTranslat
305                                 "ResourceSourceIndex, ResourceSource, Descriptor
306                                 "TranslationKeyword)",
307                                 "QWord IO Resource Descriptor macro"},
308     {"QWordMemory",             "(ResourceTypeKeyword, DecodeKeyword, MinKeyword
309                                 "MemTypeKeyword, ReadWriteKeyword, "
310                                 "AddressGranularity, AddressMinimum, AddressMaxi
311                                 "RangeLength, ResourceSourceIndex, ResourceSourc
312                                 "DescriptorName, AddressKeyword, "
313                                 "TypeKeyword)",
314                                 "QWord Memory Resource Descriptor macro"},
315     {"QWordSpace",              "(ResourceType, ResourceTypeKeyword, DecodeKeywo
316                                 "MinKeyword, MaxKeyword, TypeSpecificFlags, "
317                                 "AddressGranularity, AddressMinimum, AddressMaxi
318                                 "RangeLength, ResourceSourceIndex, ResourceSourc
319                                 "Qword Space Resource Descriptor macro"},
320     {"RawDataBuffer",           "(BufferSize) {ByteList} => RawDataBuffer",
321                                 "Create a raw data buffer (does not use Buffer A
322     {"RefOf",                   "(Object) => ObjectReference",
323                                 "Create Reference to an object"},
324     {"Register",                "(AddressSpaceKeyword, RegisterBitWidth, "
325                                 "RegisterBitOffset, RegisterAddress, "
```



```
326                                 "AccessSize, DescriptorName)",
327                                 "Generic register Resource Descriptor macro"},
328     {"Release",                 "(SyncObject)",
329                                 "Release a synchronization object"},
330     {"Reset",                   "(SyncObject)",
331                                 "Reset a synchronization object"},
332     {"ResourceTemplate",        "() {ResourceMacroList} => Buffer",
333                                 "Resource to buffer conversion macro"},
334     {"Return",                  "None | () | (ReturnArg)",
335                                 "Return from method execution"},
336     {"Revision",                "=> Integer",
337                                 "Constant revision object"},
338     {"Scope",                   "(Location) {ObjectList}",
339                                 "Open named scope "},
340     {"ShiftLeft",               "(Source, ShiftCount, Result) => Integer",
341                                 "Integer shift value left"},
342     {"ShiftRight",              "(Source, ShiftCount, Result) => Integer",
343                                 "Integer shift value right"},
344     {"Signal",                  "(SyncObject)",
345                                 "Signal a synchronization object"},
346     {"SizeOf",                  "(ObjectName) => Integer",
347                                 "Get the size of a buffer}, string}, or package"
348     {"Sleep",                   "(Milliseconds)",
349                                 "Sleep n milliseconds (yields the processor)"},
350     {"SpiSerialBus",            "(DeviceSelection, PolarityKeyword, WireModeKeyw
351                                 "DataBitLength, SlaveModeKeyword, "
352                                 "ConnectionSpeed, ClockPolarityKeyword, ClockPha
353                                 "ResourceSource, ResourceSourceIndex, "
354                                 "ResourceTypeKeyword, DescriptorName, RawDataBuf
355                                 "SPI Serial Bus Connection Resource Descriptor M
356     {"Stall",                   "(Microseconds)",
357                                 "Delay n microseconds (does not yield the proces
358     {"StartDependentFn",        "(CompatibilityPriority, PerformancePriority) {R
359                                 "Start Dependent Function Resource Descriptor ma
360     {"StartDependentFnNoPri",   "() {ResourceList}",
361                                 "Start Dependent Function Resource Descriptor ma
362     {"Store",                   "(Source, Destination) => DataRefObject",
363                                 "Store object"},
364     {"Subtract",                "(Minuend, Subtrahend, Result) => Integer",
365                                 "Integer Subtract"},
366     {"Switch",                  "(Expression) {CaseTermList}",
367                                 "Select code to execute based on expression valu
368     {"ThermalZone",             "(ThermalZoneName) {ObjectList}",
369                                 "Declare a thermal zone package"},
370     {"Timer",                   "=> Integer",
371                                 "Get 64-bit timer value"},
372     {"ToBCD",                   "(Value, Result) => Integer",
373                                 "Convert Integer to BCD"},
374     {"ToBuffer",                "(Data, Result) => Buffer",
375                                 "Convert data type to buffer"},
376     {"ToDecimalString",         "(Data, Result) => String",
377                                 "Convert data type to decimal string"},
378     {"ToHexString",             "(Data, Result) => String",
379                                 "Convert data type to hexadecimal string"},
380     {"ToInteger",               "(Data, Result) => Integer",
381                                 "Convert data type to integer"},
382     {"ToString",                "(Source, Length, Result) => String",
383                                 "Copy ASCII string from buffer"},
384     {"ToUUID",                  "(AsciiString) => Buffer",
385                                 "Convert Ascii string to UUID"},
386     {"UartSerialBus",           "(ConnectionSpeed, ByteLengthKeyword, StopBitsKe
387                                 "LinesInUse, EndianKeyword, ParityKeyword, "
388                                 "FlowControlKeyword, ReceiveBufferSize, Transmit
389                                 "ResourceSourceIndex, ResourceTypeKeyword, Descr
390                                 "RawDataBuffer() {VendorData})",
391                                 "UART Serial Bus Connection Resource Descriptor
```

```
392     {"Unicode",                 "(String) => Buffer",
393                                 "String to Unicode conversion macro"},
394     {"Unload",                  "(Handle)",
395                                 "Unload definition block"},
396     {"VendorLong",              "(DescriptorName) {VendorByteList}",
397                                 "Vendor Resource Descriptor"},
398     {"VendorShort",             "(DescriptorName) {VendorByteList}",
399                                 "Vendor Resource Descriptor"},
400     {"Wait",                    "(SyncObject, TimeoutValue) => Boolean",
401                                 "Wait on an Event"},
402     {"While",                   "(Predicate) {TermList}",
403                                 "Conditional loop"},
404     {"WordBusNumber",           "(ResourceTypeKeyword, MinKeyword, MaxKeyword, D
405                                 "AddressGranularity, AddressMinimum, "
406                                 "AddressMaximum, AddressTranslation, RangeLength
407                                 "ResourceSource, DescriptorName)",
408                                 "Word Bus number Resource Descriptor macro"},
409     {"WordIO",                  "(ResourceTypeKeyword, MinKeyword, MaxKeyword, D
410                                 "RangeTypeKeyword, AddressGranularity, "
411                                 "AddressMinimum, AddressMaximum, AddressTranslat
412                                 "ResourceSourceIndex, ResourceSource, Descriptor
413                                 "TranslationKeyword)",
414                                 "Word IO Resource Descriptor macro"},
415     {"WordSpace",               "(ResourceType, ResourceTypeKeyword, DecodeKeywo
416                                 "MaxKeyword, TypeSpecificFlags, "
417                                 "AddressGranularity, AddressMinimum, AddressMaxi
418                                 "RangeLength, ResourceSourceIndex, ResourceSourc
419                                 "Word Space Resource Descriptor macro"},
420     {"XOr",                     "(Source1, Source2, Result) => Integer",
421                                 "Integer Bitwise Xor"},
422     {"Zero",                    "=> Integer",
423                                 "Constant Zero object (0)"},
424     {NULL, NULL, NULL}
425 };
```

```
**********************************************************
   23886 Thu Dec 26 13:50:10 2013
new/usr/src/common/acpica/tools/acpihelp/ahdecode.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ahdecode - Operator/Opcode decoding for acpihelp utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define ACPI_CREATE_PREDEFINED_TABLE
#define ACPI_CREATE_RESOURCE_TABLE

#include "acpihelp.h"
#include "acpredef.h"


/* Device IDs defined in the ACPI specification */

static const AH_DEVICE_ID  AhDeviceIds[] =
{
    {"PNP0A05",     "Generic Container Device"},
    {"PNP0A06",     "Generic Container Device"},
    {"PNP0C08",     "ACPI core hardware"},
    {"PNP0C09",     "Embedded Controller Device"},
    {"PNP0C0A",     "Control Method Battery"},
    {"PNP0C0B",     "Fan"},
    {"PNP0C0C",     "Power Button Device"},
```



```
    {"PNP0C0D",     "Lid Device"},
    {"PNP0C0E",     "Sleep Button Device"},
    {"PNP0C0F",     "PCI Interrupt Link Device"},
    {"PNP0C80",     "Memory Device"},

    {"ACPI0001",    "SMBus 1.0 Host Controller"},
    {"ACPI0002",    "Smart Battery Subsystem"},
    {"ACPI0003",    "Power Source Device"},
    {"ACPI0004",    "Module Device"},
    {"ACPI0005",    "SMBus 2.0 Host Controller"},
    {"ACPI0006",    "GPE Block Device"},
    {"ACPI0007",    "Processor Device"},
    {"ACPI0008",    "Ambient Light Sensor Device"},
    {"ACPI0009",    "I/O xAPIC Device"},
    {"ACPI000A",    "I/O APIC Device"},
    {"ACPI000B",    "I/O SAPIC Device"},
    {"ACPI000C",    "Processor Aggregator Device"},
    {"ACPI000D",    "Power Meter Device"},
    {"ACPI000E",    "Time/Alarm Device"},
    {"ACPI000F",    "User Presence Detection Device"},

    {NULL, NULL}
};

#define AH_DISPLAY_EXCEPTION(Status, Name) \
    printf ("%.4X: %s\n", Status, Name)

#define AH_DISPLAY_EXCEPTION_TEXT(Status, Exception) \
    printf ("%.4X: %-28s (%s)\n", Status, Exception->Name, Exception->Descriptio

#define BUFFER_LENGTH           128
#define LINE_BUFFER_LENGTH      512

static char         Gbl_Buffer[BUFFER_LENGTH];
static char         Gbl_LineBuffer[LINE_BUFFER_LENGTH];

/* Local prototypes */

static BOOLEAN
AhDisplayPredefinedName (
    char                    *Name,
    UINT32                  Length);

static void
AhDisplayPredefinedInfo (
    char                    *Name);

static void
AhDisplayResourceName (
    const ACPI_PREDEFINED_INFO  *ThisName);

static void
AhDisplayAmlOpcode (
    const AH_AML_OPCODE     *Op);

static void
AhDisplayAslOperator (
    const AH_ASL_OPERATOR   *Op);

static void
AhDisplayOperatorKeywords (
    const AH_ASL_OPERATOR   *Op);

static void
AhDisplayAslKeyword (
    const AH_ASL_KEYWORD    *Op);
```

```
129 static void
130 AhPrintOneField (
131     UINT32                  Indent,
132     UINT32                  CurrentPosition,
133     UINT32                  MaxPosition,
134     const char              *Field);


137 /******************************************************************************
138  *
139  * FUNCTION:    AhFindPredefinedNames (entry point for predefined name search)
140  *
141  * PARAMETERS:  NamePrefix          - Name or prefix to find. Must start with
142  *                                    an underscore. NULL means "find all"
143  *
144  * RETURN:      None
145  *
146  * DESCRIPTION: Find and display all ACPI predefined names that match the
147  *              input name or prefix. Includes the required number of arguments
148  *              and the expected return type, if any.
149  *
150  *****************************************************************************/

152 void
153 AhFindPredefinedNames (
154     char                    *NamePrefix)
155 {
156     UINT32                  Length;
157     BOOLEAN                 Found;
158     char                    Name[9];


161     if (!NamePrefix)
162     {
163         Found = AhDisplayPredefinedName (Name, 0);
164         return;
165     }

167     /* Contruct a local name or name prefix */

169     AhStrupr (NamePrefix);
170     if (*NamePrefix == '_')
171     {
172         NamePrefix++;
173     }

175     Name[0] = '_';
176     strncpy (&Name[1], NamePrefix, 7);

178     Length = strlen (Name);
179     if (Length > 4)
180     {
181         printf ("%.8s: Predefined name must be 4 characters maximum\n", Name);
182         return;
183     }

185     Found = AhDisplayPredefinedName (Name, Length);
186     if (!Found)
187     {
188         printf ("%s, no matching predefined names\n", Name);
189     }
190 }


193 /******************************************************************************
```



```
194  *
195  * FUNCTION:    AhDisplayPredefinedName
196  *
197  * PARAMETERS:  Name                - Name or name prefix
198  *
199  * RETURN:      TRUE if any names matched, FALSE otherwise
200  *
201  * DESCRIPTION: Display information about ACPI predefined names that match
202  *              the input name or name prefix.
203  *
204  *****************************************************************************/

206 static BOOLEAN
207 AhDisplayPredefinedName (
208     char                    *Name,
209     UINT32                  Length)
210 {
211     const AH_PREDEFINED_NAME    *Info;
212     BOOLEAN                     Found = FALSE;
213     BOOLEAN                     Matched;
214     UINT32                      i;


217     /* Find/display all names that match the input name prefix */

219     for (Info = AslPredefinedInfo; Info->Name; Info++)
220     {
221         if (!Name)
222         {
223             Found = TRUE;
224             printf ("%s: <%s>\n", Info->Name, Info->Description);
225             printf ("%*s%s\n", 6, " ", Info->Action);

227             AhDisplayPredefinedInfo (Info->Name);
228             continue;
229         }

231         Matched = TRUE;
232         for (i = 0; i < Length; i++)
233         {
234             if (Info->Name[i] != Name[i])
235             {
236                 Matched = FALSE;
237                 break;
238             }
239         }

241         if (Matched)
242         {
243             Found = TRUE;
244             printf ("%s: <%s>\n", Info->Name, Info->Description);
245             printf ("%*s%s\n", 6, " ", Info->Action);

247             AhDisplayPredefinedInfo (Info->Name);
248         }
249     }

251     return (Found);
252 }


255 /******************************************************************************
256  *
257  * FUNCTION:    AhDisplayPredefinedInfo
258  *
259  * PARAMETERS:  Name                - Exact 4-character ACPI name.
```

```
 *
 * RETURN:      None
 *
 * DESCRIPTION: Find the name in the main ACPICA predefined info table and
 *              display the # of arguments and the return value type.
 *
 *              Note: Resource Descriptor field names do not appear in this
 *              table -- thus, nothing will be displayed for them.
 *
 ******************************************************************************/

static void
AhDisplayPredefinedInfo (
    char                        *Name)
{
    const ACPI_PREDEFINED_INFO  *ThisName;


    /* NOTE: we check both tables always because there are some dupes */

    /* Check against the predefine methods first */

    ThisName = AcpiUtMatchPredefinedMethod (Name);
    if (ThisName)
    {
        AcpiUtDisplayPredefinedMethod (Gbl_Buffer, ThisName, TRUE);
    }

    /* Check against the predefined resource descriptor names */

    ThisName = AcpiUtMatchResourceName (Name);
    if (ThisName)
    {
        AhDisplayResourceName (ThisName);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AhDisplayResourceName
 *
 * PARAMETERS:  ThisName            - Entry in the predefined method/name table
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display information about a resource descriptor name.
 *
 *****************************************************************************/

static void
AhDisplayResourceName (
    const ACPI_PREDEFINED_INFO  *ThisName)
{
    UINT32                      NumTypes;


    NumTypes = AcpiUtGetResourceBitWidth (Gbl_Buffer,
        ThisName->Info.ArgumentList);

    printf ("      %4.4s resource descriptor field is %s bits wide%s\n",
        ThisName->Info.Name,
        Gbl_Buffer,
        (NumTypes > 1) ? " (depending on descriptor type)" : "");
}
```



```
/******************************************************************************
 *
 * FUNCTION:    AhFindAmlOpcode (entry point for AML opcode name search)
 *
 * PARAMETERS:  Name                - Name or prefix for an AML opcode.
 *                                    NULL means "find all"
 *
 * RETURN:      None
 *
 * DESCRIPTION: Find all AML opcodes that match the input Name or name
 *              prefix.
 *
 *****************************************************************************/

void
AhFindAmlOpcode (
    char                    *Name)
{
    const AH_AML_OPCODE     *Op;
    BOOLEAN                 Found = FALSE;


    AhStrupr (Name);

    /* Find/display all opcode names that match the input name prefix */

    for (Op = AmlOpcodeInfo; Op->OpcodeString; Op++)
    {
        if (!Op->OpcodeName) /* Unused opcodes */
        {
            continue;
        }

        if (!Name)
        {
            AhDisplayAmlOpcode (Op);
            Found = TRUE;
            continue;
        }

        /* Upper case the opcode name before substring compare */

        strcpy (Gbl_Buffer, Op->OpcodeName);
        AhStrupr (Gbl_Buffer);

        if (strstr (Gbl_Buffer, Name) == Gbl_Buffer)
        {
            AhDisplayAmlOpcode (Op);
            Found = TRUE;
        }
    }

    if (!Found)
    {
        printf ("%s, no matching AML operators\n", Name);
    }
}


/******************************************************************************
 *
 * FUNCTION:    AhDecodeAmlOpcode (entry point for AML opcode search)
 *
 * PARAMETERS:  OpcodeString        - String version of AML opcode
 *
```

```
 * RETURN:      None
 *
 * DESCRIPTION: Display information about the input AML opcode
 *
 ******************************************************************************/

void
AhDecodeAmlOpcode (
    char                    *OpcodeString)
{
    const AH_AML_OPCODE     *Op;
    UINT32                  Opcode;
    UINT8                   Prefix;


    if (!OpcodeString)
    {
        AhFindAmlOpcode (NULL);
        return;
    }

    Opcode = ACPI_STRTOUL (OpcodeString, NULL, 16);
    if (Opcode > ACPI_UINT16_MAX)
    {
        printf ("Invalid opcode (more than 16 bits)\n");
        return;
    }

    /* Only valid opcode extension is 0x5B */

    Prefix = (Opcode & 0x0000FF00) >> 8;
    if (Prefix && (Prefix != 0x5B))
    {
        printf ("Invalid opcode (invalid extension prefix 0x%X)\n",
            Prefix);
        return;
    }

    /* Find/Display the opcode. May fall within an opcode range */

    for (Op = AmlOpcodeInfo; Op->OpcodeString; Op++)
    {
        if ((Opcode >= Op->OpcodeRangeStart) &&
            (Opcode <= Op->OpcodeRangeEnd))
        {
            AhDisplayAmlOpcode (Op);
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    AhDisplayAmlOpcode
 *
 * PARAMETERS:  Op                  - An opcode info struct
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display the contents of an AML opcode information struct
 *
 ******************************************************************************/

static void
AhDisplayAmlOpcode (
    const AH_AML_OPCODE     *Op)
```



```
{

    if (!Op->OpcodeName)
    {
        printf ("%18s: Opcode=%-9s\n", "Reserved opcode", Op->OpcodeString);
        return;
    }

    /* Opcode name and value(s) */

    printf ("%18s: Opcode=%-9s Type (%s)",
        Op->OpcodeName, Op->OpcodeString, Op->Type);

    /* Optional fixed/static arguments */

    if (Op->FixedArguments)
    {
        printf (" FixedArgs (");
        AhPrintOneField (37, 36 + 7 + strlen (Op->Type) + 12,
            AH_MAX_AML_LINE_LENGTH, Op->FixedArguments);
        printf (")");
    }

    /* Optional variable-length argument list */

    if (Op->VariableArguments)
    {
        if (Op->FixedArguments)
        {
            printf ("\n%*s", 36, " ");
        }
        printf (" VariableArgs (");
        AhPrintOneField (37, 15, AH_MAX_AML_LINE_LENGTH, Op->VariableArguments);
        printf (")");
    }
    printf ("\n");

    /* Grammar specification */

    if (Op->Grammar)
    {
        AhPrintOneField (37, 0, AH_MAX_AML_LINE_LENGTH, Op->Grammar);
        printf ("\n");
    }
}


/******************************************************************************
 *
 * FUNCTION:    AhFindAslKeywords (entry point for ASL keyword search)
 *
 * PARAMETERS:  Name                - Name or prefix for an ASL keyword.
 *                                    NULL means "find all"
 *
 * RETURN:      None
 *
 * DESCRIPTION: Find all ASL keywords that match the input Name or name
 *              prefix.
 *
 ******************************************************************************/

void
AhFindAslKeywords (
    char                    *Name)
{
    const AH_ASL_KEYWORD    *Keyword;
```

```
524     BOOLEAN                 Found = FALSE;


527     AhStrupr (Name);

529     for (Keyword = AslKeywordInfo; Keyword->Name; Keyword++)
530     {
531         if (!Name)
532         {
533             AhDisplayAslKeyword (Keyword);
534             Found = TRUE;
535             continue;
536         }

538         /* Upper case the operator name before substring compare */

540         strcpy (Gbl_Buffer, Keyword->Name);
541         AhStrupr (Gbl_Buffer);

543         if (strstr (Gbl_Buffer, Name) == Gbl_Buffer)
544         {
545             AhDisplayAslKeyword (Keyword);
546             Found = TRUE;
547         }
548     }

550     if (!Found)
551     {
552         printf ("%s, no matching ASL keywords\n", Name);
553     }
554 }


557 /******************************************************************************
558  *
559  * FUNCTION:    AhDisplayAslKeyword
560  *
561  * PARAMETERS:  Op                  - Pointer to ASL keyword with syntax info
562  *
563  * RETURN:      None
564  *
565  * DESCRIPTION: Format and display syntax info for an ASL keyword. Splits
566  *              long lines appropriately for reading.
567  *
568  *****************************************************************************/

570 static void
571 AhDisplayAslKeyword (
572     const AH_ASL_KEYWORD    *Op)
573 {

575     /* ASL keyword name and description */

577     printf ("%22s: %s\n", Op->Name, Op->Description);
578     if (!Op->KeywordList)
579     {
580         return;
581     }

583     /* List of actual keywords */

585     AhPrintOneField (24, 0, AH_MAX_ASL_LINE_LENGTH, Op->KeywordList);
586     printf ("\n");
587 }
```



```
590 /******************************************************************************
591  *
592  * FUNCTION:    AhFindAslOperators (entry point for ASL operator search)
593  *
594  * PARAMETERS:  Name                - Name or prefix for an ASL operator.
595  *                                    NULL means "find all"
596  *
597  * RETURN:      None
598  *
599  * DESCRIPTION: Find all ASL operators that match the input Name or name
600  *              prefix.
601  *
602  *****************************************************************************/

604 void
605 AhFindAslOperators (
606     char                    *Name)
607 {
608     const AH_ASL_OPERATOR   *Operator;
609     BOOLEAN                 Found = FALSE;


612     AhStrupr (Name);

614     /* Find/display all names that match the input name prefix */

616     for (Operator = AslOperatorInfo; Operator->Name; Operator++)
617     {
618         if (!Name)
619         {
620             AhDisplayAslOperator (Operator);
621             Found = TRUE;
622             continue;
623         }

625         /* Upper case the operator name before substring compare */

627         strcpy (Gbl_Buffer, Operator->Name);
628         AhStrupr (Gbl_Buffer);

630         if (strstr (Gbl_Buffer, Name) == Gbl_Buffer)
631         {
632             AhDisplayAslOperator (Operator);
633             Found = TRUE;
634         }
635     }

637     if (!Found)
638     {
639         printf ("%s, no matching ASL operators\n", Name);
640     }
641 }


644 /******************************************************************************
645  *
646  * FUNCTION:    AhDisplayAslOperator
647  *
648  * PARAMETERS:  Op                  - Pointer to ASL operator with syntax info
649  *
650  * RETURN:      None
651  *
652  * DESCRIPTION: Format and display syntax info for an ASL operator. Splits
653  *              long lines appropriately for reading.
654  *
655  *****************************************************************************/
```

```
static void
AhDisplayAslOperator (
    const AH_ASL_OPERATOR   *Op)
{

    /* ASL operator name and description */

    printf ("%16s: %s\n", Op->Name, Op->Description);
    if (!Op->Syntax)
    {
        return;
    }

    /* Syntax for the operator */

    AhPrintOneField (18, 0, AH_MAX_ASL_LINE_LENGTH, Op->Syntax);
    printf ("\n");

    AhDisplayOperatorKeywords (Op);
    printf ("\n");
}


/*******************************************************************************
 *
 * FUNCTION:    AhDisplayOperatorKeywords
 *
 * PARAMETERS:  Op                  - Pointer to ASL keyword with syntax info
 *
 * RETURN:      None
 *
 * DESCRIPTION: Display any/all keywords that are associated with the ASL
 *              operator.
 *
 ******************************************************************************/

static void
AhDisplayOperatorKeywords (
    const AH_ASL_OPERATOR   *Op)
{
    char                    *Token;
    char                    *Separators = "(){}, ";
    BOOLEAN                 FirstKeyword = TRUE;


    if (!Op || !Op->Syntax)
    {
        return;
    }

    /*
     * Find all parameters that have the word "keyword" within, and then
     * display the info about that keyword
     */
    strcpy (Gbl_LineBuffer, Op->Syntax);
    Token = strtok (Gbl_LineBuffer, Separators);
    while (Token)
    {
        if (strstr (Token, "Keyword"))
        {
            if (FirstKeyword)
            {
                printf ("\n");
                FirstKeyword = FALSE;
            }
```



```
            /* Found a keyword, display keyword information */

            AhFindAslKeywords (Token);
        }

        Token = strtok (NULL, Separators);
    }
}


/*******************************************************************************
 *
 * FUNCTION:    AhPrintOneField
 *
 * PARAMETERS:  Indent              - Indent length for new line(s)
 *              CurrentPosition     - Position on current line
 *              MaxPosition         - Max allowed line length
 *              Field               - Data to output
 *
 * RETURN:      Line position after field is written
 *
 * DESCRIPTION: Split long lines appropriately for ease of reading.
 *
 ******************************************************************************/

static void
AhPrintOneField (
    UINT32                  Indent,
    UINT32                  CurrentPosition,
    UINT32                  MaxPosition,
    const char              *Field)
{
    UINT32                  Position;
    UINT32                  TokenLength;
    const char              *This;
    const char              *Next;
    const char              *Last;


    This = Field;
    Position = CurrentPosition;

    if (Position == 0)
    {
        printf ("%*s", (int) Indent, " ");
        Position = Indent;
    }

    Last = This + strlen (This);
    while ((Next = strpbrk (This, " ")))
    {
        TokenLength = Next - This;
        Position += TokenLength;

        /* Split long lines */

        if (Position > MaxPosition)
        {
            printf ("\n%*s", (int) Indent, " ");
            Position = TokenLength;
        }

        printf ("%.*s ", (int) TokenLength, This);
        This = Next + 1;
    }
```

```
789     /* Handle last token on the input line */

791     TokenLength = Last - This;
792     if (TokenLength > 0)
793     {
794         Position += TokenLength;
795         if (Position > MaxPosition)
796         {
797             printf ("\n%*s", (int) Indent, " ");
798         }
799         printf ("%s", This);
800     }
801 }


804 /******************************************************************************
805  *
806  * FUNCTION:    AhDisplayDeviceIds
807  *
808  * PARAMETERS:  None
809  *
810  * RETURN:      None
811  *
812  * DESCRIPTION: Display all PNP* and ACPI* device IDs defined in the ACPI spec.
813  *
814  *****************************************************************************/

816 void
817 AhDisplayDeviceIds (
818     void)
819 {
820     const AH_DEVICE_ID      *DeviceId = AhDeviceIds;


823     printf ("ACPI and PNP Device IDs defined in the ACPI specification:\n\n");
824     while (DeviceId->Name)
825     {
826         printf ("%8s   %s\n", DeviceId->Name, DeviceId->Description);
827         DeviceId++;
828     }
829 }


832 /******************************************************************************
833  *
834  * FUNCTION:    AhDecodeException
835  *
836  * PARAMETERS:  HexString           - ACPI status string from command line, in
837  *                                    hex. If null, display all exceptions.
838  *
839  * RETURN:      None
840  *
841  * DESCRIPTION: Decode and display an ACPI_STATUS exception code.
842  *
843  *****************************************************************************/

845 void
846 AhDecodeException (
847     char                    *HexString)
848 {
849     const ACPI_EXCEPTION_INFO   *ExceptionInfo;
850     UINT32                      Status;
851     UINT32                      i;
```



```
854     /*
855      * A null input string means to decode and display all known
856      * exception codes.
857      */
858     if (!HexString)
859     {
860         printf ("All defined ACPICA exception codes:\n\n");
861         AH_DISPLAY_EXCEPTION (0, "AE_OK                         (No error occurre

863         /* Display codes in each block of exception types */

865         for (i = 1; (i & AE_CODE_MASK) <= AE_CODE_MAX; i += 0x1000)
866         {
867             Status = i;
868             do
869             {
870                 ExceptionInfo = AcpiUtValidateException ((ACPI_STATUS) Status);
871                 if (ExceptionInfo)
872                 {
873                     AH_DISPLAY_EXCEPTION_TEXT (Status, ExceptionInfo);
874                 }
875                 Status++;

877             } while (ExceptionInfo);
878         }
879         return;
880     }

882     /* Decode a single user-supplied exception code */

884     Status = ACPI_STRTOUL (HexString, NULL, 16);
885     if (!Status)
886     {
887         printf ("%s: Invalid hexadecimal exception code value\n", HexString);
888         return;
889     }

891     if (Status > ACPI_UINT16_MAX)
892     {
893         AH_DISPLAY_EXCEPTION (Status, "Invalid exception code (more than 16 bits
894         return;
895     }

897     ExceptionInfo = AcpiUtValidateException ((ACPI_STATUS) Status);
898     if (!ExceptionInfo)
899     {
900         AH_DISPLAY_EXCEPTION (Status, "Unknown exception code");
901         return;
902     }

904     AH_DISPLAY_EXCEPTION_TEXT (Status, ExceptionInfo);
905 }
```

```
**********************************************************
    7002 Thu Dec 26 13:50:10 2013
new/usr/src/common/acpica/tools/acpihelp/ahmain.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ahmain - Main module for the acpi help utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpihelp.h"


/* Local prototypes */

static void
AhDisplayUsage (
    void);

#define AH_UTILITY_NAME             "ACPI Help Utility"
#define AH_SUPPORTED_OPTIONS        "ehikmopsv"


/******************************************************************************
 *
 * FUNCTION:    AhDisplayUsage
 *
 * DESCRIPTION: Usage message
```



```
 *
 *****************************************************************************/

static void
AhDisplayUsage (
    void)
{

    ACPI_USAGE_HEADER ("acpihelp <options> [NamePrefix | HexValue]");
    ACPI_OPTION ("-h",                          "Display help");
    ACPI_OPTION ("-v",                          "Display version information");

    printf ("\nACPI Names and Symbols:\n");
    ACPI_OPTION ("-k [NamePrefix]",         "Find/Display ASL non-operator keywo
    ACPI_OPTION ("-m [NamePrefix]",         "Find/Display AML opcode name(s)");
    ACPI_OPTION ("-p [NamePrefix]",         "Find/Display ASL predefined method
    ACPI_OPTION ("-s [NamePrefix]",         "Find/Display ASL operator name(s)")

    printf ("\nACPI Values:\n");
    ACPI_OPTION ("-e [HexValue]",           "Decode ACPICA exception code");
    ACPI_OPTION ("-i",                      "Display known ACPI Device IDs (_HID
    ACPI_OPTION ("-o [HexValue]",           "Decode hex AML opcode");

    printf ("\nNamePrefix/HexValue not specified means \"Display All\"\n");
    printf ("\nDefault search with NamePrefix and no options:\n");
    printf ("    Find ASL operator names - if NamePrefix does not start with und
    printf ("    Find ASL predefined method names - if NamePrefix starts with un
}


/******************************************************************************
 *
 * FUNCTION:    main
 *
 * DESCRIPTION: C main function for AcpiHelp utility.
 *
 *****************************************************************************/

int ACPI_SYSTEM_XFACE
main (
    int                     argc,
    char                    *argv[])
{
    char                    *Name;
    UINT32                  DecodeType;
    int                     j;


    ACPI_DEBUG_INITIALIZE (); /* For debug version only */
    printf (ACPI_COMMON_SIGNON (AH_UTILITY_NAME));
    DecodeType = AH_DECODE_DEFAULT;

    if (argc < 2)
    {
        AhDisplayUsage ();
        return (0);
    }

    /* Command line options */

    while ((j = AcpiGetopt (argc, argv, AH_SUPPORTED_OPTIONS)) != EOF) switch (j
    {
    case 'e':

        DecodeType = AH_DECODE_EXCEPTION;
        break;
```

```
129     case 'i':

131         DecodeType = AH_DISPLAY_DEVICE_IDS;
132         break;

134     case 'k':

136         DecodeType = AH_DECODE_ASL_KEYWORD;
137         break;

139     case 'm':

141         DecodeType = AH_DECODE_AML;
142         break;

144     case 'o':

146         DecodeType = AH_DECODE_AML_OPCODE;
147         break;

149     case 'p':

151         DecodeType = AH_DECODE_PREDEFINED_NAME;
152         break;

154     case 's':

156         DecodeType = AH_DECODE_ASL;
157         break;

159     case 'v': /* -v: (Version): signon already emitted, just exit */

161         return (0);

163     case 'h':
164     default:

166         AhDisplayUsage ();
167         return (-1);
168     }

170     /* Missing (null) name means "display all" */

172     Name = argv[AcpiGbl_Optind];

174     switch (DecodeType)
175     {
176     case AH_DECODE_AML:

178         AhFindAmlOpcode (Name);
179         break;

181     case AH_DECODE_AML_OPCODE:

183         AhDecodeAmlOpcode (Name);
184         break;

186     case AH_DECODE_PREDEFINED_NAME:

188         AhFindPredefinedNames (Name);
189         break;

191     case AH_DECODE_ASL:

193         AhFindAslOperators (Name);
```



```
194         break;

196     case AH_DECODE_ASL_KEYWORD:

198         AhFindAslKeywords (Name);
199         break;

201     case AH_DISPLAY_DEVICE_IDS:

203         AhDisplayDeviceIds ();
204         break;

206     case AH_DECODE_EXCEPTION:

208         AhDecodeException (Name);
209         break;

211     default:

213         if (!Name)
214         {
215             AhFindAslOperators (Name);
216             break;
217         }

219         if (*Name == '_')
220         {
221             AhFindPredefinedNames (Name);
222         }
223         else
224         {
225             AhFindAslOperators (Name);
226         }
227         break;
228     }

230     return (0);
231 }


234 /*******************************************************************************
235  *
236  * FUNCTION:    AhStrupr (strupr)
237  *
238  * PARAMETERS:  SrcString           - The source string to convert
239  *
240  * RETURN:      None
241  *
242  * DESCRIPTION: Convert string to uppercase
243  *
244  * NOTE: This is not a POSIX function, so it appears here, not in utclib.c
245  *
246  ******************************************************************************/

248 void
249 AhStrupr (
250     char                    *SrcString)
251 {
252     char                    *String;


255     if (!SrcString)
256     {
257         return;
258     }
```

```
260     /* Walk entire string, uppercasing the letters */

262     for (String = SrcString; *String; String++)
263     {
264         *String = (char) toupper ((int) *String);
265     }

267     return;
268 }
```

```
**********************************************************
    2317 Thu Dec 26 13:50:11 2013
new/usr/src/common/acpica/tools/acpinames/acpinames.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acpinames.h - Common include for AcpiNames utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _ACPINAMES_H
#define _ACPINAMES_H

#include "acpi.h"
#include "accommon.h"
#include "acapps.h"
#include "../acpiexec/aecommon.h"

#include <stdio.h>

#define ACPI_MAX_INIT_TABLES    (32)

#endif
```

```
**********************************************************
    8211 Thu Dec 26 13:50:11 2013
new/usr/src/common/acpica/tools/acpinames/anmain.c
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /******************************************************************************
   2  *
   3  * Module Name: anmain - Main routine for the AcpiNames utility
   4  *
   5  *****************************************************************************/

   7 /*
   8  * Copyright (C) 2000 - 2013, Intel Corp.
   9  * All rights reserved.
  10  *
  11  * Redistribution and use in source and binary forms, with or without
  12  * modification, are permitted provided that the following conditions
  13  * are met:
  14  * 1. Redistributions of source code must retain the above copyright
  15  *    notice, this list of conditions, and the following disclaimer,
  16  *    without modification.
  17  * 2. Redistributions in binary form must reproduce at minimum a disclaimer
  18  *    substantially similar to the "NO WARRANTY" disclaimer below
  19  *    ("Disclaimer") and any redistribution must be conditioned upon
  20  *    including a substantially similar Disclaimer requirement for further
  21  *    binary redistribution.
  22  * 3. Neither the names of the above-listed copyright holders nor the names
  23  *    of any contributors may be used to endorse or promote products derived
  24  *    from this software without specific prior written permission.
  25  *
  26  * Alternatively, this software may be distributed under the terms of the
  27  * GNU General Public License ("GPL") version 2 as published by the Free
  28  * Software Foundation.
  29  *
  30  * NO WARRANTY
  31  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  32  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  33  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
  34  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  35  * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  36  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  37  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  38  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  39  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  40  * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  41  * POSSIBILITY OF SUCH DAMAGES.
  42  */

  44 #include "acpinames.h"

  46 #define _COMPONENT          ACPI_TOOLS
  47         ACPI_MODULE_NAME    ("anmain")


  50 extern ACPI_TABLE_DESC  Tables[];

  52 static AE_TABLE_DESC    *AeTableListHead = NULL;


  55 #define AN_UTILITY_NAME             "ACPI Namespace Dump Utility"
  56 #define AN_SUPPORTED_OPTIONS        "?hv"


  59 /******************************************************************************
```



```
  60  *
  61  * FUNCTION:    usage
  62  *
  63  * PARAMETERS:  None
  64  *
  65  * RETURN:      None
  66  *
  67  * DESCRIPTION: Print a usage message
  68  *
  69  *****************************************************************************/

  71 static void
  72 usage (
  73     void)
  74 {

  76     ACPI_USAGE_HEADER ("AcpiNames [options] AMLfile");
  77     ACPI_OPTION ("-?",                  "Display this message");
  78     ACPI_OPTION ("-v",                  "Display version information");
  79 }


  82 /******************************************************************************
  83  *
  84  * FUNCTION:    NsDumpEntireNamespace
  85  *
  86  * PARAMETERS:  AmlFilename         - Filename for DSDT or SSDT AML table
  87  *
  88  * RETURN:      Status (pass/fail)
  89  *
  90  * DESCRIPTION: Build an ACPI namespace for the input AML table, and dump the
  91  *              formatted namespace contents.
  92  *
  93  *****************************************************************************/

  95 static int
  96 NsDumpEntireNamespace (
  97     char                    *AmlFilename)
  98 {
  99     ACPI_STATUS             Status;
 100     ACPI_TABLE_HEADER       *Table = NULL;
 101     UINT32                  TableCount = 0;
 102     AE_TABLE_DESC           *TableDesc;
 103     ACPI_HANDLE             Handle;


 106     /* Open the binary AML file and read the entire table */

 108     Status = AcpiDbReadTableFromFile (AmlFilename, &Table);
 109     if (ACPI_FAILURE (Status))
 110     {
 111         printf ("**** Could not get input table %s, %s\n", AmlFilename,
 112             AcpiFormatException (Status));
 113         return (-1);
 114     }

 116     /* Table must be a DSDT. SSDTs are not currently supported */

 118     if (!ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_DSDT))
 119     {
 120         printf ("**** Input table signature is [%4.4s], must be [DSDT]\n",
 121             Table->Signature);
 122         return (-1);
 123     }

 125     /*
```

```
126      * Allocate and link a table descriptor (allows for future expansion to
127      * multiple input files)
128      */
129     TableDesc = AcpiOsAllocate (sizeof (AE_TABLE_DESC));
130     TableDesc->Table = Table;
131     TableDesc->Next = AeTableListHead;
132     AeTableListHead = TableDesc;

134     TableCount++;

136     /*
137      * Build a local XSDT with all tables. Normally, here is where the
138      * RSDP search is performed to find the ACPI tables
139      */
140     Status = AeBuildLocalTables (TableCount, AeTableListHead);
141     if (ACPI_FAILURE (Status))
142     {
143         return (-1);
144     }

146     /* Initialize table manager, get XSDT */

148     Status = AcpiInitializeTables (Tables, ACPI_MAX_INIT_TABLES, TRUE);
149     if (ACPI_FAILURE (Status))
150     {
151         printf ("**** Could not initialize ACPI table manager, %s\n",
152             AcpiFormatException (Status));
153         return (-1);
154     }

156     /* Reallocate root table to dynamic memory */

158     Status = AcpiReallocateRootTable ();
159     if (ACPI_FAILURE (Status))
160     {
161         printf ("**** Could not reallocate root table, %s\n",
162             AcpiFormatException (Status));
163         return (-1);
164     }

166     /* Load the ACPI namespace */

168     Status = AcpiLoadTables ();
169     if (ACPI_FAILURE (Status))
170     {
171         printf ("**** Could not load ACPI tables, %s\n",
172             AcpiFormatException (Status));
173         return (-1);
174     }

176     /*
177      * Enable ACPICA. These calls don't do much for this
178      * utility, since we only dump the namespace. There is no
179      * hardware or event manager code underneath.
180      */
181     Status = AcpiEnableSubsystem (
182                 ACPI_NO_ACPI_ENABLE |
183                 ACPI_NO_ADDRESS_SPACE_INIT |
184                 ACPI_NO_EVENT_INIT |
185                 ACPI_NO_HANDLER_INIT);
186     if (ACPI_FAILURE (Status))
187     {
188         printf ("**** Could not EnableSubsystem, %s\n",
189             AcpiFormatException (Status));
190         return (-1);
191     }
```



```
193     Status = AcpiInitializeObjects (
194                 ACPI_NO_ADDRESS_SPACE_INIT |
195                 ACPI_NO_DEVICE_INIT |
196                 ACPI_NO_EVENT_INIT);
197     if (ACPI_FAILURE (Status))
198     {
199         printf ("**** Could not InitializeObjects, %s\n",
200             AcpiFormatException (Status));
201         return (-1);
202     }

204     /*
205      * Perform a namespace walk to dump the contents
206      */
207     AcpiOsPrintf ("\nACPI Namespace:\n");

209     AcpiNsDumpObjects (ACPI_TYPE_ANY, ACPI_DISPLAY_SUMMARY, ACPI_UINT32_MAX,
210         ACPI_OWNER_ID_MAX, AcpiGbl_RootNode);


213     /* Example: get a handle to the _GPE scope */

215     Status = AcpiGetHandle (NULL, "\\_GPE", &Handle);
216     AE_CHECK_OK (AcpiGetHandle, Status);

218     return (0);
219 }


222 /******************************************************************************
223  *
224  * FUNCTION:    main
225  *
226  * PARAMETERS:  argc, argv
227  *
228  * RETURN:      Status (pass/fail)
229  *
230  * DESCRIPTION: Main routine for NsDump utility
231  *
232  *****************************************************************************/

234 int ACPI_SYSTEM_XFACE
235 main (
236     int                     argc,
237     char                    **argv)
238 {
239     ACPI_STATUS             Status;
240     int                     j;


243     ACPI_DEBUG_INITIALIZE (); /* For debug version only */
244     printf (ACPI_COMMON_SIGNON (AN_UTILITY_NAME));

246     if (argc < 2)
247     {
248         usage ();
249         return (0);
250     }

252     /* Init globals and ACPICA */

254     AcpiDbgLevel = ACPI_NORMAL_DEFAULT | ACPI_LV_TABLES;
255     AcpiDbgLayer = 0xFFFFFFFF;

257     Status = AcpiInitializeSubsystem ();
```

```
258     AE_CHECK_OK (AcpiInitializeSubsystem, Status);
259
260     /* Get the command line options */
261
262     while ((j = AcpiGetopt (argc, argv, AN_SUPPORTED_OPTIONS)) != EOF) switch(j)
263     {
264     case 'v': /* -v: (Version): signon already emitted, just exit */
265
266         return (0);
267
268     case '?':
269     case 'h':
270     default:
271
272         usage();
273         return (0);
274     }
275
276     /*
277      * The next argument is the filename for the DSDT or SSDT.
278      * Open the file, build namespace and dump it.
279      */
280     return (NsDumpEntireNamespace (argv[AcpiGbl_Optind]));
281 }
```

```
**********************************************************
    8820 Thu Dec 26 13:50:11 2013
new/usr/src/common/acpica/tools/acpinames/anstubs.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: anstubs - Stub routines for the AcpiNames utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpinames.h"

#include <acutils.h>
#include <acevents.h>
#include <acdispat.h>

#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("anstubs")


/******************************************************************************
 *
 * DESCRIPTION: Stubs used to facilitate linkage of the NsDump utility.
 *
 *****************************************************************************/
```



```
/* Utilities */

void
AcpiUtSubsystemShutdown (
    void)
{
}

ACPI_STATUS
AcpiUtExecute_STA (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    UINT32                  *StatusFlags)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_HID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_CID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID_LIST **ReturnCidList)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_UID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_SUB (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecutePowerMethods (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    const char              **MethodNames,
    UINT8                   MethodCount,
    UINT8                   *OutValues)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtEvaluateNumericObject (
    char                    *ObjectName,
    ACPI_NAMESPACE_NODE     *DeviceNode,
    UINT64                  *Value)
{
    return (AE_NOT_IMPLEMENTED);
}
```

```
ACPI_STATUS
AcpiUtCopyIobjectToEobject (
    ACPI_OPERAND_OBJECT     *Obj,
    ACPI_BUFFER             *RetBuffer)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtCopyEobjectToIobject (
    ACPI_OBJECT             *Obj,
    ACPI_OPERAND_OBJECT     **InternalObj)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtCopyIobjectToIobject (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     **DestDesc,
    ACPI_WALK_STATE         *WalkState)
{
    return (AE_NOT_IMPLEMENTED);
}


/* Hardware manager */

UINT32
AcpiHwGetMode (
    void)
{
    return (0);
}


/* Event manager */

ACPI_STATUS
AcpiEvInstallXruptHandlers (
    void)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiEvInitializeEvents (
    void)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiEvInstallRegionHandlers (
    void)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiEvInitializeOpRegions (
    void)
{
    return (AE_OK);
}
```



```
ACPI_STATUS
AcpiEvInitializeRegion (
    ACPI_OPERAND_OBJECT     *RegionObj,
    BOOLEAN                 AcpiNsLocked)
{
    return (AE_OK);
}

#if (!ACPI_REDUCED_HARDWARE)
ACPI_STATUS
AcpiEvDeleteGpeBlock (
    ACPI_GPE_BLOCK_INFO     *GpeBlock)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiEnable (
    void)
{
    return (AE_OK);
}
#endif /* !ACPI_REDUCED_HARDWARE */


/* AML Interpreter */

void
AcpiExUnlinkMutex (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
}

void
AcpiExReleaseAllMutexes (
    ACPI_THREAD_STATE       *Thread)
{
}

ACPI_STATUS
AcpiExReadDataFromField (
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     **RetBufferDesc)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiExWriteDataToField (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_OPERAND_OBJECT     **ResultDesc)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiExPrepFieldValue (
    ACPI_CREATE_FIELD_INFO  *Info)
{
    return (AE_OK);
}

ACPI_STATUS
```

```
AcpiExAcquireMutexObject (
    UINT16                  Timeout,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_THREAD_ID          ThreadId)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiExReleaseMutexObject (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiExStoreObjectToNode (
    ACPI_OPERAND_OBJECT     *SourceDesc,
    ACPI_NAMESPACE_NODE     *Node,
    ACPI_WALK_STATE         *WalkState,
    UINT8                   ImplicitConversion)
{
    return (AE_NOT_IMPLEMENTED);
}


/* Namespace manager */

ACPI_STATUS
AcpiNsEvaluate (
    ACPI_EVALUATE_INFO      *Info)
{
    return (AE_NOT_IMPLEMENTED);
}

void
AcpiNsExecModuleCodeList (
    void)
{
}


/* Dispatcher */

ACPI_STATUS
AcpiDsInitializeObjects (
    UINT32                  TableIndex,
    ACPI_NAMESPACE_NODE     *StartNode)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsCallControlMethod (
    ACPI_THREAD_STATE       *Thread,
    ACPI_WALK_STATE         *WalkState,
    ACPI_PARSE_OBJECT       *Op)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiDsRestartControlMethod (
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     *ReturnDesc)
{
```



```
    return (AE_NOT_IMPLEMENTED);
}

void
AcpiDsTerminateControlMethod (
    ACPI_OPERAND_OBJECT     *MethodDesc,
    ACPI_WALK_STATE         *WalkState)
{
}

ACPI_STATUS
AcpiDsMethodError (
    ACPI_STATUS             Status,
    ACPI_WALK_STATE         *WalkState)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiDsBeginMethodExecution (
    ACPI_NAMESPACE_NODE     *MethodNode,
    ACPI_OPERAND_OBJECT     *ObjDesc,
    ACPI_WALK_STATE         *WalkState)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiDsGetPredicateValue (
    ACPI_WALK_STATE         *WalkState,
    ACPI_OPERAND_OBJECT     *ResultObj)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiDsGetBufferFieldArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsGetBankFieldArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsGetRegionArguments (
    ACPI_OPERAND_OBJECT     *RgnDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsGetBufferArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc)
{
    return (AE_OK);
}

ACPI_STATUS
AcpiDsGetPackageArguments (
    ACPI_OPERAND_OBJECT     *ObjDesc)
```

```
391 {
392     return (AE_OK);
393 }

395 ACPI_STATUS
396 AcpiDsExecBeginOp (
397     ACPI_WALK_STATE         *WalkState,
398     ACPI_PARSE_OBJECT       **OutOp)
399 {
400     return (AE_NOT_IMPLEMENTED);
401 }

403 ACPI_STATUS
404 AcpiDsExecEndOp (
405     ACPI_WALK_STATE         *State)
406 {
407     return (AE_NOT_IMPLEMENTED);
408 }


411 /* AML Debugger */

413 void
414 AcpiDbDisplayArgumentObject (
415     ACPI_OPERAND_OBJECT     *ObjDesc,
416     ACPI_WALK_STATE         *WalkState)
417 {
418 }

420 ACPI_STATUS
421 AcpiDbInitialize (
422     void)
423 {
424     return (AE_OK);
425 }

427 void
428 AcpiDbTerminate (
429     void)
430 {
431 }

433 /* OSL interfaces */

435 ACPI_THREAD_ID
436 AcpiOsGetThreadId (
437     void)
438 {
439     return (0xFFFF);
440 }

442 ACPI_STATUS
443 AcpiOsExecute (
444     ACPI_EXECUTE_TYPE       Type,
445     ACPI_OSD_EXEC_CALLBACK  Function,
446     void                    *Context)
447 {
448     return (AE_SUPPORT);
449 }
```

```
**********************************************************
    9903 Thu Dec 26 13:50:11 2013
new/usr/src/common/acpica/tools/acpinames/antables.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: antables - ACPI table setup/install for AcpiNames utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpinames.h"

#define _COMPONENT          ACPI_TOOLS
        ACPI_MODULE_NAME    ("antables")

/* Local prototypes */

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);

/* Non-AML tables that are constructed locally and installed */

static ACPI_TABLE_RSDP          LocalRSDP;
static ACPI_TABLE_FACS          LocalFACS;

/*
```



```
 * We need a local FADT so that the hardware subcomponent will function,
 * even though the underlying OSD HW access functions don't do anything.
 */
static ACPI_TABLE_FADT          LocalFADT;

/*
 * Use XSDT so that both 32- and 64-bit versions of this utility will
 * function automatically.
 */
static ACPI_TABLE_XSDT          *LocalXSDT;

#define BASE_XSDT_TABLES        1
#define BASE_XSDT_SIZE          (sizeof (ACPI_TABLE_XSDT) + \
                                    ((BASE_XSDT_TABLES -1) * sizeof (UINT64)))

ACPI_TABLE_DESC                 Tables[ACPI_MAX_INIT_TABLES];


/******************************************************************************
 *
 * FUNCTION:    AeBuildLocalTables
 *
 * PARAMETERS:  TableCount      - Number of tables on the command line
 *              TableList       - List of actual tables from files
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Build a complete ACPI table chain, with a local RSDP, XSDT,
 *              FADT, FACS, and the input DSDT/SSDT.
 *
 *****************************************************************************/

ACPI_STATUS
AeBuildLocalTables (
    UINT32                  TableCount,
    AE_TABLE_DESC           *TableList)
{
    ACPI_PHYSICAL_ADDRESS   DsdtAddress = 0;
    UINT32                  XsdtSize;
    AE_TABLE_DESC           *NextTable;
    UINT32                  NextIndex;
    ACPI_TABLE_FADT         *ExternalFadt = NULL;


    /*
     * Update the table count. For DSDT, it is not put into the XSDT. For
     * FADT, this is already accounted for since we usually install a
     * local FADT.
     */
    NextTable = TableList;
    while (NextTable)
    {
        if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT) ||
            ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
        {
            TableCount--;
        }
        NextTable = NextTable->Next;
    }

    XsdtSize = BASE_XSDT_SIZE + (TableCount * sizeof (UINT64));

    /* Build an XSDT */

    LocalXSDT = AcpiOsAllocate (XsdtSize);
    if (!LocalXSDT)
```

```
127     {
128         return (AE_NO_MEMORY);
129     }

131     ACPI_MEMSET (LocalXSDT, 0, XsdtSize);
132     ACPI_MOVE_NAME (LocalXSDT->Header.Signature, ACPI_SIG_XSDT);
133     LocalXSDT->Header.Length = XsdtSize;
134     LocalXSDT->Header.Revision = 1;

136     LocalXSDT->TableOffsetEntry[0] = ACPI_PTR_TO_PHYSADDR (&LocalFADT);

138     /*
139      * Install the user tables. The DSDT must be installed in the FADT.
140      * All other tables are installed directly into the XSDT.
141      */
142     NextIndex = BASE_XSDT_TABLES;
143     NextTable = TableList;
144     while (NextTable)
145     {
146         /*
147          * Incoming DSDT or FADT are special cases. All other tables are
148          * just immediately installed into the XSDT.
149          */
150         if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT))
151         {
152             if (DsdtAddress)
153             {
154                 printf ("Already found a DSDT, only one allowed\n");
155                 return (AE_ALREADY_EXISTS);
156             }

158             /* The incoming user table is a DSDT */

160             DsdtAddress = ACPI_PTR_TO_PHYSADDR (NextTable->Table);
161         }
162         else if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
163         {
164             ExternalFadt = ACPI_CAST_PTR (ACPI_TABLE_FADT, NextTable->Table);
165             LocalXSDT->TableOffsetEntry[2] = ACPI_PTR_TO_PHYSADDR (NextTable->Ta
166         }
167         else
168         {
169             /* Install the table in the XSDT */

171             LocalXSDT->TableOffsetEntry[NextIndex] = ACPI_PTR_TO_PHYSADDR (NextT
172             NextIndex++;
173         }

175         NextTable = NextTable->Next;
176     }

178     /* Build an RSDP */

180     ACPI_MEMSET (&LocalRSDP, 0, sizeof (ACPI_TABLE_RSDP));
181     ACPI_MAKE_RSDP_SIG (LocalRSDP.Signature);
182     ACPI_MEMCPY (LocalRSDP.OemId, "I_TEST", 6);
183     LocalRSDP.Revision = 2;
184     LocalRSDP.XsdtPhysicalAddress = ACPI_PTR_TO_PHYSADDR (LocalXSDT);
185     LocalRSDP.Length = sizeof (ACPI_TABLE_XSDT);

187     /* Set checksums for both XSDT and RSDP */

189     LocalXSDT->Header.Checksum = (UINT8) -AcpiTbChecksum (
190         (void *) LocalXSDT, LocalXSDT->Header.Length);
191     LocalRSDP.Checksum = (UINT8) -AcpiTbChecksum (
192         (void *) &LocalRSDP, ACPI_RSDP_CHECKSUM_LENGTH);
```



```
194     if (!DsdtAddress)
195     {
196         return (AE_SUPPORT);
197     }

199     if (ExternalFadt)
200     {
201         /*
202          * Use the external FADT, but we must update the DSDT/FACS addresses
203          * as well as the checksum
204          */
205         ExternalFadt->Dsdt = DsdtAddress;
206         ExternalFadt->Facs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);

208         if (ExternalFadt->Header.Length > ACPI_PTR_DIFF (&ExternalFadt->XDsdt, E
209         {
210             ExternalFadt->XDsdt = DsdtAddress;
211             ExternalFadt->XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
212         }
213         /* Complete the FADT with the checksum */

215         ExternalFadt->Header.Checksum = 0;
216         ExternalFadt->Header.Checksum = (UINT8) -AcpiTbChecksum (
217             (void *) ExternalFadt, ExternalFadt->Header.Length);
218     }
219     else
220     {
221         /*
222          * Build a local FADT so we can test the hardware/event init
223          */
224         ACPI_MEMSET (&LocalFADT, 0, sizeof (ACPI_TABLE_FADT));
225         ACPI_MOVE_NAME (LocalFADT.Header.Signature, ACPI_SIG_FADT);

227         /* Setup FADT header and DSDT/FACS addresses */

229         LocalFADT.Dsdt = 0;
230         LocalFADT.Facs = 0;

232         LocalFADT.XDsdt = DsdtAddress;
233         LocalFADT.XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);

235         LocalFADT.Header.Revision = 3;
236         LocalFADT.Header.Length = sizeof (ACPI_TABLE_FADT);

238         /* Miscellaneous FADT fields */

240         LocalFADT.Gpe0BlockLength = 16;
241         LocalFADT.Gpe0Block = 0x00001234;

243         LocalFADT.Gpe1BlockLength = 6;
244         LocalFADT.Gpe1Block = 0x00005678;
245         LocalFADT.Gpe1Base = 96;

247         LocalFADT.Pm1EventLength = 4;
248         LocalFADT.Pm1aEventBlock = 0x00001aaa;
249         LocalFADT.Pm1bEventBlock = 0x00001bbb;

251         LocalFADT.Pm1ControlLength = 2;
252         LocalFADT.Pm1aControlBlock = 0xB0;

254         LocalFADT.PmTimerLength = 4;
255         LocalFADT.PmTimerBlock = 0xA0;

257         LocalFADT.Pm2ControlBlock = 0xC0;
258         LocalFADT.Pm2ControlLength = 1;
```

```
260         /* Setup one example X-64 field */

262         LocalFADT.XPm1bEventBlock.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;
263         LocalFADT.XPm1bEventBlock.Address = LocalFADT.Pm1bEventBlock;
264         LocalFADT.XPm1bEventBlock.BitWidth = (UINT8) ACPI_MUL_8 (LocalFADT.Pm1Ev

266         /* Complete the FADT with the checksum */

268         LocalFADT.Header.Checksum = 0;
269         LocalFADT.Header.Checksum = (UINT8) -AcpiTbChecksum (
270             (void *) &LocalFADT, LocalFADT.Header.Length);
271     }

273     /* Build a FACS */

275     ACPI_MEMSET (&LocalFACS, 0, sizeof (ACPI_TABLE_FACS));
276     ACPI_MOVE_NAME (LocalFACS.Signature, ACPI_SIG_FACS);

278     LocalFACS.Length = sizeof (ACPI_TABLE_FACS);
279     LocalFACS.GlobalLock = 0x11AA0011;

281     return (AE_OK);
282 }


285 /******************************************************************************
286  *
287  * FUNCTION:    AeLocalGetRootPointer
288  *
289  * PARAMETERS:  None
290  *
291  * RETURN:      Address of the RSDP
292  *
293  * DESCRIPTION: Return a local RSDP, used to dynamically load tables via the
294  *              standard ACPI mechanism.
295  *
296  *****************************************************************************/

298 ACPI_PHYSICAL_ADDRESS
299 AeLocalGetRootPointer (
300     void)
301 {

303     return ((ACPI_PHYSICAL_ADDRESS) &LocalRSDP);
304 }
```

```
**********************************************************
   12438 Thu Dec 26 13:50:11 2013
new/usr/src/common/acpica/tools/acpisrc/acpisrc.h
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: acpisrc.h - Include file for AcpiSrc utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"

#include <stdio.h>
#include <sys/stat.h>
#include <errno.h>

/* mkdir support */

#ifdef WIN32
#include <direct.h>
#else
#define mkdir(x) mkdir(x, 0770)
#endif


/* Constants */
```



```
#define LINES_IN_LEGAL_HEADER               105 /* See above */
#define LEGAL_HEADER_SIGNATURE              " * 2.1. This is your license from I
#define LINES_IN_LINUX_HEADER               34
#define LINUX_HEADER_SIGNATURE              " * THIS SOFTWARE IS PROVIDED BY THE
#define LINES_IN_ASL_HEADER                 29 /* Header as output from disassem

#define ASRC_MAX_FILE_SIZE                  (1024 * 100)

#define FILE_TYPE_SOURCE                    1
#define FILE_TYPE_HEADER                    2
#define FILE_TYPE_DIRECTORY                 3

#define CVT_COUNT_TABS                      0x00000001
#define CVT_COUNT_NON_ANSI_COMMENTS         0x00000002
#define CVT_TRIM_LINES                      0x00000004
#define CVT_CHECK_BRACES                    0x00000008
#define CVT_COUNT_LINES                     0x00000010
#define CVT_BRACES_ON_SAME_LINE             0x00000020
#define CVT_MIXED_CASE_TO_UNDERSCORES       0x00000040
#define CVT_LOWER_CASE_IDENTIFIERS          0x00000080
#define CVT_REMOVE_DEBUG_MACROS             0x00000100
#define CVT_TRIM_WHITESPACE                 0x00000200  /* Should be after all l
#define CVT_REMOVE_EMPTY_BLOCKS             0x00000400  /* Should be after trimm
#define CVT_REDUCE_TYPEDEFS                 0x00000800
#define CVT_COUNT_SHORTMULTILINE_COMMENTS   0x00001000
#define CVT_SPACES_TO_TABS4                 0x40000000  /* Tab conversion should
#define CVT_SPACES_TO_TABS8                 0x80000000  /* Tab conversion should

#define FLG_DEFAULT_FLAGS                   0x00000000
#define FLG_NO_CARRIAGE_RETURNS             0x00000001
#define FLG_NO_FILE_OUTPUT                  0x00000002
#define FLG_LOWERCASE_DIRNAMES              0x00000004

#define AS_START_IGNORE                     "/*!"
#define AS_STOP_IGNORE                      "!*/"


/* Globals */

extern UINT32                   Gbl_Files;
extern UINT32                   Gbl_MissingBraces;
extern UINT32                   Gbl_Tabs;
extern UINT32                   Gbl_NonAnsiComments;
extern UINT32                   Gbl_SourceLines;
extern UINT32                   Gbl_WhiteLines;
extern UINT32                   Gbl_CommentLines;
extern UINT32                   Gbl_LongLines;
extern UINT32                   Gbl_TotalLines;
extern UINT32                   Gbl_HeaderSize;
extern UINT32                   Gbl_HeaderLines;
extern struct stat              Gbl_StatBuf;
extern char                     *Gbl_FileBuffer;
extern UINT32                   Gbl_TotalSize;
extern UINT32                   Gbl_FileSize;
extern UINT32                   Gbl_FileType;
extern BOOLEAN                  Gbl_VerboseMode;
extern BOOLEAN                  Gbl_QuietMode;
extern BOOLEAN                  Gbl_BatchMode;
extern BOOLEAN                  Gbl_MadeChanges;
extern BOOLEAN                  Gbl_Overwrite;
extern BOOLEAN                  Gbl_WidenDeclarations;
extern BOOLEAN                  Gbl_IgnoreLoneLineFeeds;
extern BOOLEAN                  Gbl_HasLoneLineFeeds;
extern BOOLEAN                  Gbl_Cleanup;
extern BOOLEAN                  Gbl_IgnoreTranslationEscapes;
```

```
127 extern void                     *Gbl_StructDefs;

129 #define PARAM_LIST(pl)          pl
130 #define TERSE_PRINT(a)          if (!Gbl_VerboseMode) printf PARAM_LIST(a)
131 #define VERBOSE_PRINT(a)        if (Gbl_VerboseMode) printf PARAM_LIST(a)

133 #define REPLACE_WHOLE_WORD      0x00
134 #define REPLACE_SUBSTRINGS      0x01
135 #define REPLACE_MASK            0x01

137 #define EXTRA_INDENT_C          0x02


140 /* Conversion table structs */

142 typedef struct acpi_string_table
143 {
144     char                        *Target;
145     char                        *Replacement;
146     UINT8                       Type;

148 } ACPI_STRING_TABLE;


151 typedef struct acpi_typed_identifier_table
152 {
153     char                        *Identifier;
154     UINT8                       Type;

156 } ACPI_TYPED_IDENTIFIER_TABLE;

158 #define SRC_TYPE_SIMPLE         0
159 #define SRC_TYPE_STRUCT         1
160 #define SRC_TYPE_UNION          2


163 typedef struct acpi_identifier_table
164 {
165     char                        *Identifier;

167 } ACPI_IDENTIFIER_TABLE;

169 typedef struct acpi_conversion_table
170 {
171     char                        *NewHeader;
172     UINT32                      Flags;

174     ACPI_TYPED_IDENTIFIER_TABLE *LowerCaseTable;

176     ACPI_STRING_TABLE           *SourceStringTable;
177     ACPI_IDENTIFIER_TABLE       *SourceLineTable;
178     ACPI_IDENTIFIER_TABLE       *SourceConditionalTable;
179     ACPI_IDENTIFIER_TABLE       *SourceMacroTable;
180     ACPI_TYPED_IDENTIFIER_TABLE *SourceStructTable;
181     ACPI_IDENTIFIER_TABLE       *SourceSpecialMacroTable;
182     UINT32                      SourceFunctions;

184     ACPI_STRING_TABLE           *HeaderStringTable;
185     ACPI_IDENTIFIER_TABLE       *HeaderLineTable;
186     ACPI_IDENTIFIER_TABLE       *HeaderConditionalTable;
187     ACPI_IDENTIFIER_TABLE       *HeaderMacroTable;
188     ACPI_TYPED_IDENTIFIER_TABLE *HeaderStructTable;
189     ACPI_IDENTIFIER_TABLE       *HeaderSpecialMacroTable;
190     UINT32                      HeaderFunctions;

192 } ACPI_CONVERSION_TABLE;
```



```
195 /* Conversion tables */

197 extern ACPI_CONVERSION_TABLE       LinuxConversionTable;
198 extern ACPI_CONVERSION_TABLE       CleanupConversionTable;
199 extern ACPI_CONVERSION_TABLE       StatsConversionTable;
200 extern ACPI_CONVERSION_TABLE       CustomConversionTable;
201 extern ACPI_CONVERSION_TABLE       LicenseConversionTable;
202 extern ACPI_CONVERSION_TABLE       IndentConversionTable;


205 /* Prototypes */

207 char *
208 AsSkipUntilChar (
209     char                        *Buffer,
210     char                        Target);

212 char *
213 AsSkipPastChar (
214     char                        *Buffer,
215     char                        Target);

217 char *
218 AsReplaceData (
219     char                        *Buffer,
220     UINT32                      LengthToRemove,
221     char                        *BufferToAdd,
222     UINT32                      LengthToAdd);

224 int
225 AsReplaceString (
226     char                        *Target,
227     char                        *Replacement,
228     UINT8                       Type,
229     char                        *Buffer);

231 int
232 AsLowerCaseString (
233     char                        *Target,
234     char                        *Buffer);

236 void
237 AsRemoveLine (
238     char                        *Buffer,
239     char                        *Keyword);

241 void
242 AsRemoveMacro (
243     char                        *Buffer,
244     char                        *Keyword);

246 void
247 AsCheckForBraces (
248     char                        *Buffer,
249     char                        *Filename);

251 void
252 AsTrimLines (
253     char                        *Buffer,
254     char                        *Filename);

256 void
257 AsMixedCaseToUnderscores (
258     char                        *Buffer,
```

```
259     char                    *Filename);

261 void
262 AsCountTabs (
263     char                    *Buffer,
264     char                    *Filename);

266 void
267 AsBracesOnSameLine (
268     char                    *Buffer);

270 void
271 AsLowerCaseIdentifiers (
272     char                    *Buffer);

274 void
275 AsReduceTypedefs (
276     char                    *Buffer,
277     char                    *Keyword);

279 void
280 AsRemoveDebugMacros (
281     char                    *Buffer);

283 void
284 AsRemoveEmptyBlocks (
285     char                    *Buffer,
286     char                    *Filename);

288 void
289 AsCleanupSpecialMacro (
290     char                    *Buffer,
291     char                    *Keyword);

293 void
294 AsCountSourceLines (
295     char                    *Buffer,
296     char                    *Filename);

298 void
299 AsCountNonAnsiComments (
300     char                    *Buffer,
301     char                    *Filename);

303 void
304 AsTrimWhitespace (
305     char                    *Buffer);

307 void
308 AsTabify4 (
309     char                    *Buffer);

311 void
312 AsTabify8 (
313     char                    *Buffer);

315 void
316 AsRemoveConditionalCompile (
317     char                    *Buffer,
318     char                    *Keyword);

320 ACPI_NATIVE_INT
321 AsProcessTree (
322     ACPI_CONVERSION_TABLE   *ConversionTable,
323     char                    *SourcePath,
324     char                    *TargetPath);
```



```
326 int
327 AsGetFile (
328     char                    *FileName,
329     char                    **FileBuffer,
330     UINT32                  *FileSize);

332 int
333 AsPutFile (
334     char                    *Pathname,
335     char                    *FileBuffer,
336     UINT32                  SystemFlags);

338 void
339 AsReplaceHeader (
340     char                    *Buffer,
341     char                    *NewHeader);

343 void
344 AsConvertFile (
345     ACPI_CONVERSION_TABLE   *ConversionTable,
346     char                    *FileBuffer,
347     char                    *Filename,
348     ACPI_NATIVE_INT         FileType);

350 ACPI_NATIVE_INT
351 AsProcessOneFile (
352     ACPI_CONVERSION_TABLE   *ConversionTable,
353     char                    *SourcePath,
354     char                    *TargetPath,
355     int                     MaxPathLength,
356     char                    *Filename,
357     ACPI_NATIVE_INT         FileType);

359 ACPI_NATIVE_INT
360 AsCheckForDirectory (
361     char                    *SourceDirPath,
362     char                    *TargetDirPath,
363     char                    *Filename,
364     char                    **SourcePath,
365     char                    **TargetPath);

367 void
368 AsRemoveExtraLines (
369     char                    *FileBuffer,
370     char                    *Filename);

372 void
373 AsRemoveSpacesAfterPeriod (
374     char                    *FileBuffer,
375     char                    *Filename);

377 BOOLEAN
378 AsMatchExactWord (
379     char                    *Word,
380     UINT32                  WordLength);

382 void
383 AsPrint (
384     char                    *Message,
385     UINT32                  Count,
386     char                    *Filename);

388 void
389 AsInsertPrefix (
390     char                    *Buffer,
```

```
391     char                    *Keyword,
392     UINT8                   Type);

394 char *
395 AsInsertData (
396     char                    *Buffer,
397     char                    *BufferToAdd,
398     UINT32                  LengthToAdd);

400 char *
401 AsRemoveData (
402     char                    *StartPointer,
403     char                    *EndPointer);

405 void
406 AsInsertCarriageReturns (
407     char                    *Buffer);

409 void
410 AsConvertToLineFeeds (
411     char                    *Buffer);

413 void
414 AsStrlwr (
415     char                    *SrcString);
```

```
**********************************************************
   18586 Thu Dec 26 13:50:12 2013
new/usr/src/common/acpica/tools/acpisrc/ascase.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: ascase - Source conversion - lower/upper case utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpisrc.h"

/* Local prototypes */

void
AsUppercaseTokens (
    char                    *Buffer,
    char                    *PrefixString);


/******************************************************************************
 *
 * FUNCTION:    AsLowerCaseString
 *
 * DESCRIPTION: LowerCase all instances of a target string with a replacement
 *              string. Returns count of the strings replaced.
 *
```



```
 *****************************************************************************/

int
AsLowerCaseString (
    char                    *Target,
    char                    *Buffer)
{
    char                    *SubString1;
    char                    *SubString2;
    char                    *SubBuffer;
    int                     TargetLength;
    int                     LowerCaseCount = 0;
    int                     i;


    TargetLength = strlen (Target);

    SubBuffer = Buffer;
    SubString1 = Buffer;

    while (SubString1)
    {
        /* Find the target string */

        SubString1 = strstr (SubBuffer, Target);
        if (!SubString1)
        {
            return (LowerCaseCount);
        }

        /*
         * Check for translation escape string -- means to ignore
         * blocks of code while replacing
         */
        if (Gbl_IgnoreTranslationEscapes)
        {
            SubString2 = NULL;
        }
        else
        {
            SubString2 = strstr (SubBuffer, AS_START_IGNORE);
        }

        if ((SubString2) &&
            (SubString2 < SubString1))
        {
            /* Find end of the escape block starting at "Substring2" */

            SubString2 = strstr (SubString2, AS_STOP_IGNORE);
            if (!SubString2)
            {
                /* Didn't find terminator */

                return (LowerCaseCount);
            }

            /* Move buffer to end of escape block and continue */

            SubBuffer = SubString2;
        }

        /* Do the actual replace if the target was found */

        else
        {
            if (!AsMatchExactWord (SubString1, TargetLength))
```

```
127             {
128                 SubBuffer = SubString1 + 1;
129                 continue;
130             }

132             for (i = 0; i < TargetLength; i++)
133             {
134                 SubString1[i] = (char) tolower ((int) SubString1[i]);
135             }

137             SubBuffer = SubString1 + TargetLength;

139             if ((Gbl_WidenDeclarations) && (!Gbl_StructDefs))
140             {
141                 if ((SubBuffer[0] == ' ') && (SubBuffer[1] == ' '))
142                 {
143                     AsInsertData (SubBuffer, "        ", 8);
144                 }
145             }

147             LowerCaseCount++;
148         }
149     }

151     return (LowerCaseCount);
152 }


155 /******************************************************************************
156  *
157  * FUNCTION:    AsMixedCaseToUnderscores
158  *
159  * DESCRIPTION: Converts mixed case identifiers to underscored identifiers.
160  *              for example,
161  *
162  *              ThisUsefullyNamedIdentifier   becomes:
163  *
164  *              this_usefully_named_identifier
165  *
166  *****************************************************************************/

168 void
169 AsMixedCaseToUnderscores (
170     char                    *Buffer,
171     char                    *Filename)
172 {
173     UINT32                  Length;
174     char                    *SubBuffer = Buffer;
175     char                    *TokenEnd;
176     char                    *TokenStart = NULL;
177     char                    *SubString;
178     UINT32                  LineNumber = 1;
179     UINT32                  Count;


182     /*
183      * Examine the entire buffer (contains the entire file)
184      * We are only interested in these tokens:
185      *      Escape sequences - ignore entire sequence
186      *      Single-quoted constants - ignore
187      *      Quoted strings - ignore entire string
188      *      Translation escape - starts with /,*,!
189      *      Decimal and hex numeric constants - ignore entire token
190      *      Entire uppercase token - ignore, it is a macro or define
191      *      Starts with underscore, then a lowercase or digit: convert
192      */
```



```
193     while (*SubBuffer)
194     {
195         if (*SubBuffer == '\n')
196         {
197             LineNumber++;
198             SubBuffer++;
199             continue;
200         }

202         /* Ignore standard escape sequences (\n, \r, etc.)  Not Hex or Octal esc

204         if (*SubBuffer == '\\')
205         {
206             SubBuffer += 2;
207             continue;
208         }

210         /* Ignore single-quoted characters */

212         if (*SubBuffer == '\'')
213         {
214             SubBuffer += 3;
215             continue;
216         }

218         /* Ignore standard double-quoted strings */

220         if (*SubBuffer == '"')
221         {
222             SubBuffer++;
223             Count = 0;
224             while (*SubBuffer != '"')
225             {
226                 Count++;
227                 if ((!*SubBuffer) ||
228                     (Count > 8192))
229                 {
230                     printf ("Found an unterminated quoted string!, line %u: %s\n
231                         LineNumber, Filename);
232                     return;
233                 }

235                 /* Handle escape sequences */

237                 if (*SubBuffer == '\\')
238                 {
239                     SubBuffer++;
240                 }

242                 SubBuffer++;
243             }
244             SubBuffer++;
245             continue;
246         }

248         /*
249          * Check for translation escape string. It means to ignore
250          * blocks of code during this code conversion.
251          */
252         if ((SubBuffer[0] == '/') &&
253             (SubBuffer[1] == '*') &&
254             (SubBuffer[2] == '!'))
255         {
256             SubBuffer = strstr (SubBuffer, "!*/");
257             if (!SubBuffer)
258             {
```

```
259                 printf ("Found an unterminated translation escape!, line %u: %s\
260                     LineNumber, Filename);
261                 return;
262             }
263             continue;
264         }

266         /* Ignore anything that starts with a number (0-9) */

268         if (isdigit ((int) *SubBuffer))
269         {
270             /* Ignore hex constants */

272             if ((SubBuffer[0] == '0') &&
273                 ((SubBuffer[1] == 'x') || (SubBuffer[1] == 'X')))
274             {
275                 SubBuffer += 2;
276             }

278             /* Skip over all digits, both decimal and hex */

280             while (isxdigit ((int) *SubBuffer))
281             {
282                 SubBuffer++;
283             }
284             TokenStart = NULL;
285             continue;
286         }

288         /*
289          * Check for fully upper case identifiers. These are usually macros
290          * or defines. Allow decimal digits and embedded underscores.
291          */
292         if (isupper ((int) *SubBuffer))
293         {
294             SubString = SubBuffer + 1;
295             while ((isupper ((int) *SubString)) ||
296                    (isdigit ((int) *SubString)) ||
297                    (*SubString == '_'))
298             {
299                 SubString++;
300             }

302             /*
303              * For the next character, anything other than a lower case
304              * means that the identifier has terminated, and contains
305              * exclusively Uppers/Digits/Underscores. Ignore the entire
306              * identifier.
307              */
308             if (!islower ((int) *SubString))
309             {
310                 SubBuffer = SubString + 1;
311                 continue;
312             }
313         }

315         /*
316          * These forms may indicate an identifier that can be converted:
317          *      <UpperCase><LowerCase> (Ax)
318          *      <UpperCase><Number> (An)
319          */
320         if (isupper ((int) SubBuffer[0]) &&
321           ((islower ((int) SubBuffer[1])) || isdigit ((int) SubBuffer[1])))
322         {
323             TokenStart = SubBuffer;
324             SubBuffer++;
```



```
326             while (1)
327             {
328                 /* Walk over the lower case letters and decimal digits */

330                 while (islower ((int) *SubBuffer) ||
331                        isdigit ((int) *SubBuffer))
332                 {
333                     SubBuffer++;
334                 }

336                 /* Check for end of line or end of token */

338                 if (*SubBuffer == '\n')
339                 {
340                     LineNumber++;
341                     break;
342                 }

344                 if (*SubBuffer == ' ')
345                 {
346                     /* Check for form "Axx - " in a parameter header description

348                     while (*SubBuffer == ' ')
349                     {
350                         SubBuffer++;
351                     }

353                     SubBuffer--;
354                     if ((SubBuffer[1] == '-') &&
355                         (SubBuffer[2] == ' '))
356                     {
357                         if (TokenStart)
358                         {
359                             *TokenStart = (char) tolower ((int) *TokenStart);
360                         }
361                     }
362                     break;
363                 }

365                 /*
366                  * Ignore these combinations:
367                  *      <Letter><Digit><UpperCase>
368                  *      <Digit><Digit><UpperCase>
369                  *      <Underscore><Digit><UpperCase>
370                  */
371                 if (isdigit ((int) *SubBuffer))
372                 {
373                     if (isalnum ((int) *(SubBuffer-1)) ||
374                         *(SubBuffer-1) == '_')
375                     {
376                         break;
377                     }
378                 }

380                 /* Ignore token if next character is not uppercase or digit */

382                 if (!isupper ((int) *SubBuffer) &&
383                     !isdigit ((int) *SubBuffer))
384                 {
385                     break;
386                 }

388                 /*
389                  * Form <UpperCase><LowerCaseLetters><UpperCase> (AxxB):
390                  * Convert leading character of the token to lower case
```

```
391                  */
392                 if (TokenStart)
393                 {
394                     *TokenStart = (char) tolower ((int) *TokenStart);
395                     TokenStart = NULL;
396                 }

398                 /* Find the end of this identifier (token) */

400                 TokenEnd = SubBuffer - 1;
401                 while ((isalnum ((int) *TokenEnd)) ||
402                        (*TokenEnd == '_'))
403                 {
404                     TokenEnd++;
405                 }

407                 SubString = TokenEnd;
408                 Length = 0;

410                 while (*SubString != '\n')
411                 {
412                     /*
413                      * If we have at least two trailing spaces, we can get rid o
414                      * one to make up for the newly inserted underscore. This wi
415                      * help preserve the alignment of the text
416                      */
417                     if ((SubString[0] == ' ') &&
418                         (SubString[1] == ' '))
419                     {
420                         Length = SubString - SubBuffer - 1;
421                         break;
422                     }

424                     SubString++;
425                 }

427                 if (!Length)
428                 {
429                     Length = strlen (&SubBuffer[0]);
430                 }

432                 /*
433                  * Within this identifier, convert this pair of letters that
434                  * matches the form:
435                  *
436                  *      <LowerCase><UpperCase>
437                  * to
438                  *      <LowerCase><Underscore><LowerCase>
439                  */
440                 Gbl_MadeChanges = TRUE;

442                 /* Insert the underscore */

444                 memmove (&SubBuffer[1], &SubBuffer[0], Length + 1);
445                 SubBuffer[0] = '_';

447                 /*
448                  * If we have <UpperCase><UpperCase>, leave them as-is
449                  * Enables transforms like:
450                  *      LocalFADT -> local_FADT
451                  */
452                 if (isupper ((int) SubBuffer[2]))
453                 {
454                     SubBuffer += 1;
455                     break;
456                 }
```



```
458                 /* Lower case the original upper case letter */

460                 SubBuffer[1] = (char) tolower ((int) SubBuffer[1]);
461                 SubBuffer += 2;
462             }
463         }

465         SubBuffer++;
466     }
467 }


470 /*******************************************************************************
471  *
472  * FUNCTION:    AsLowerCaseIdentifiers
473  *
474  * DESCRIPTION: Converts mixed case identifiers to lower case. Leaves comments,
475  *              quoted strings, and all-upper-case macros alone.
476  *
477  ******************************************************************************/

479 void
480 AsLowerCaseIdentifiers (
481     char                    *Buffer)
482 {
483     char                    *SubBuffer = Buffer;


486     while (*SubBuffer)
487     {
488         /*
489          * Check for translation escape string -- means to ignore
490          * blocks of code while replacing
491          */
492         if ((SubBuffer[0] == '/') &&
493             (SubBuffer[1] == '*') &&
494             (SubBuffer[2] == '!'))
495         {
496             SubBuffer = strstr (SubBuffer, "!*/");
497             if (!SubBuffer)
498             {
499                 return;
500             }
501         }

503         /* Ignore comments */

505         if ((SubBuffer[0] == '/') &&
506             (SubBuffer[1] == '*'))
507         {
508             SubBuffer = strstr (SubBuffer, "*/");
509             if (!SubBuffer)
510             {
511                 return;
512             }

514             SubBuffer += 2;
515         }

517         /* Ignore quoted strings */

519         if ((SubBuffer[0] == '\"') && (SubBuffer[1] != '\''))
520         {
521             SubBuffer++;
```

```
523             /* Find the closing quote */

525             while (SubBuffer[0])
526             {
527                 /* Ignore escaped quote characters */

529                 if (SubBuffer[0] == '\\')
530                 {
531                     SubBuffer++;
532                 }
533                 else if (SubBuffer[0] == '\"')
534                 {
535                     SubBuffer++;
536                     break;
537                 }
538                 SubBuffer++;
539             }
540         }

542         if (!SubBuffer[0])
543         {
544             return;
545         }

547         /*
548          * Only lower case if we have an upper followed by a lower
549          * This leaves the all-uppercase things (macros, etc.) intact
550          */
551         if ((isupper ((int) SubBuffer[0])) &&
552             (islower ((int) SubBuffer[1])))
553         {
554             Gbl_MadeChanges = TRUE;
555             *SubBuffer = (char) tolower ((int) *SubBuffer);
556         }

558         SubBuffer++;
559     }
560 }


563 /*******************************************************************************
564  *
565  * FUNCTION:    AsUppercaseTokens
566  *
567  * DESCRIPTION: Force to uppercase all tokens that begin with the prefix string.
568  *              used to convert mixed-case macros and constants to uppercase.
569  *
570  ******************************************************************************/

572 void
573 AsUppercaseTokens (
574     char                    *Buffer,
575     char                    *PrefixString)
576 {
577     char                    *SubBuffer;
578     char                    *TokenEnd;
579     char                    *SubString;
580     int                     i;
581     UINT32                  Length;


584     SubBuffer = Buffer;

586     while (SubBuffer)
587     {
588         SubBuffer = strstr (SubBuffer, PrefixString);
```



```
589         if (SubBuffer)
590         {
591             TokenEnd = SubBuffer;
592             while ((isalnum ((int) *TokenEnd)) || (*TokenEnd == '_'))
593             {
594                 TokenEnd++;
595             }

597             for (i = 0; i < (TokenEnd - SubBuffer); i++)
598             {
599                 if ((islower ((int) SubBuffer[i])) &&
600                     (isupper ((int) SubBuffer[i+1])))
601                 {

603                     SubString = TokenEnd;
604                     Length = 0;

606                     while (*SubString != '\n')
607                     {
608                         if ((SubString[0] == ' ') &&
609                             (SubString[1] == ' '))
610                         {
611                             Length = SubString - &SubBuffer[i] - 2;
612                             break;
613                         }

615                         SubString++;
616                     }

618                     if (!Length)
619                     {
620                         Length = strlen (&SubBuffer[i+1]);
621                     }

623                     memmove (&SubBuffer[i+2], &SubBuffer[i+1], (Length+1));
624                     SubBuffer[i+1] = '_';
625                     i +=2;
626                     TokenEnd++;
627                 }
628             }

630             for (i = 0; i < (TokenEnd - SubBuffer); i++)
631             {
632                 SubBuffer[i] = (char) toupper ((int) SubBuffer[i]);
633             }

635             SubBuffer = TokenEnd;
636         }
637     }
638 }
```

```
**********************************************************
   39065 Thu Dec 26 13:50:12 2013
new/usr/src/common/acpica/tools/acpisrc/asconvrt.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asconvrt - Source conversion code
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpisrc.h"

/* Local prototypes */

char *
AsCheckAndSkipLiterals (
    char                    *Buffer,
    UINT32                  *TotalLines);

UINT32
AsCountLines (
    char                    *Buffer,
    char                    *Filename);

/* Opening signature of the Intel legal header */

char        *HeaderBegin = "/******************************************************
```



```
/******************************************************************************
 *
 * FUNCTION:    AsRemoveExtraLines
 *
 * DESCRIPTION: Remove all extra lines at the start and end of the file.
 *
 *****************************************************************************/

void
AsRemoveExtraLines (
    char                    *FileBuffer,
    char                    *Filename)
{
    char                    *FileEnd;
    int                     Length;


    /* Remove any extra lines at the start of the file */

    while (*FileBuffer == '\n')
    {
        printf ("Removing extra line at start of file: %s\n", Filename);
        AsRemoveData (FileBuffer, FileBuffer + 1);
    }

    /* Remove any extra lines at the end of the file */

    Length = strlen (FileBuffer);
    FileEnd = FileBuffer + (Length - 2);

    while (*FileEnd == '\n')
    {
        printf ("Removing extra line at end of file: %s\n", Filename);
        AsRemoveData (FileEnd, FileEnd + 1);
        FileEnd--;
    }
}


/******************************************************************************
 *
 * FUNCTION:    AsRemoveSpacesAfterPeriod
 *
 * DESCRIPTION: Remove an extra space after a period.
 *
 *****************************************************************************/

void
AsRemoveSpacesAfterPeriod (
    char                    *FileBuffer,
    char                    *Filename)
{
    int                     ReplaceCount = 0;
    char                    *Possible;


    Possible = FileBuffer;
    while (Possible)
    {
        Possible = strstr (Possible, ".  ");
        if (Possible)
        {
            if ((*(Possible -1) == '.')  ||
                (*(Possible -1) == '\"') ||
```

```
127                 (*(Possible -1) == '\n'))
128             {
129                 Possible += 3;
130                 continue;
131             }
132
133             Possible = AsReplaceData (Possible, 3, ". ", 2);
134             ReplaceCount++;
135         }
136     }
137
138     if (ReplaceCount)
139     {
140         printf ("Removed %d extra blanks after a period: %s\n",
141             ReplaceCount, Filename);
142     }
143 }


146 /******************************************************************************
147  *
148  * FUNCTION:    AsMatchExactWord
149  *
150  * DESCRIPTION: Check previous and next characters for whitespace
151  *
152  *****************************************************************************/

154 BOOLEAN
155 AsMatchExactWord (
156     char                    *Word,
157     UINT32                  WordLength)
158 {
159     char                    NextChar;
160     char                    PrevChar;


163     NextChar = Word[WordLength];
164     PrevChar = * (Word -1);

166     if (isalnum ((int) NextChar) ||
167         (NextChar == '_')  ||
168         isalnum ((int) PrevChar) ||
169         (PrevChar == '_'))
170     {
171         return (FALSE);
172     }

174     return (TRUE);
175 }


178 /******************************************************************************
179  *
180  * FUNCTION:    AsPrint
181  *
182  * DESCRIPTION: Common formatted print
183  *
184  *****************************************************************************/

186 void
187 AsPrint (
188     char                    *Message,
189     UINT32                  Count,
190     char                    *Filename)
191 {
```



```
193     if (Gbl_QuietMode)
194     {
195         return;
196     }

198     printf ("-- %4u %28.28s : %s\n", Count, Message, Filename);
199 }


202 /******************************************************************************
203  *
204  * FUNCTION:    AsCheckAndSkipLiterals
205  *
206  * DESCRIPTION: Generic routine to skip comments and quoted string literals.
207  *              Keeps a line count.
208  *
209  *****************************************************************************/

211 char *
212 AsCheckAndSkipLiterals (
213     char                    *Buffer,
214     UINT32                  *TotalLines)
215 {
216     UINT32                  NewLines = 0;
217     char                    *SubBuffer = Buffer;
218     char                    *LiteralEnd;


221     /* Ignore comments */

223     if ((SubBuffer[0] == '/') &&
224         (SubBuffer[1] == '*'))
225     {
226         LiteralEnd = strstr (SubBuffer, "*/");
227         SubBuffer += 2;     /* Get past comment opening */

229         if (!LiteralEnd)
230         {
231             return (SubBuffer);
232         }

234         while (SubBuffer < LiteralEnd)
235         {
236             if (*SubBuffer == '\n')
237             {
238                 NewLines++;
239             }

241             SubBuffer++;
242         }

244         SubBuffer += 2;     /* Get past comment close */
245     }

247     /* Ignore quoted strings */

249     else if (*SubBuffer == '\"')
250     {
251         SubBuffer++;
252         LiteralEnd = AsSkipPastChar (SubBuffer, '\"');
253         if (!LiteralEnd)
254         {
255             return (SubBuffer);
256         }
257     }
```

```
259     if (TotalLines)
260     {
261         (*TotalLines) += NewLines;
262     }
263     return (SubBuffer);
264 }


267 /******************************************************************************
268  *
269  * FUNCTION:    AsAsCheckForBraces
270  *
271  * DESCRIPTION: Check for an open brace after each if statement
272  *
273  *****************************************************************************/

275 void
276 AsCheckForBraces (
277     char                    *Buffer,
278     char                    *Filename)
279 {
280     char                    *SubBuffer = Buffer;
281     char                    *NextBrace;
282     char                    *NextSemicolon;
283     char                    *NextIf;
284     UINT32                  TotalLines = 1;


287     while (*SubBuffer)
288     {

290         SubBuffer = AsCheckAndSkipLiterals (SubBuffer, &TotalLines);

292         if (*SubBuffer == '\n')
293         {
294             TotalLines++;
295         }
296         else if (!(strncmp (" if", SubBuffer, 3)))
297         {
298             SubBuffer += 2;
299             NextBrace = strstr (SubBuffer, "{");
300             NextSemicolon = strstr (SubBuffer, ";");
301             NextIf = strstr (SubBuffer, " if");

303             if ((!NextBrace) ||
304                (NextSemicolon && (NextBrace > NextSemicolon)) ||
305                (NextIf && (NextBrace > NextIf)))
306             {
307                 Gbl_MissingBraces++;

309                 if (!Gbl_QuietMode)
310                 {
311                     printf ("Missing braces for <if>, line %u: %s\n", TotalLines
312                 }
313             }
314         }
315         else if (!(strncmp (" else if", SubBuffer, 8)))
316         {
317             SubBuffer += 7;
318             NextBrace = strstr (SubBuffer, "{");
319             NextSemicolon = strstr (SubBuffer, ";");
320             NextIf = strstr (SubBuffer, " if");

322             if ((!NextBrace) ||
323                (NextSemicolon && (NextBrace > NextSemicolon)) ||
324                (NextIf && (NextBrace > NextIf)))
```



```
325             {
326                 Gbl_MissingBraces++;

328                 if (!Gbl_QuietMode)
329                 {
330                     printf ("Missing braces for <if>, line %u: %s\n", TotalLines
331                 }
332             }
333         }
334         else if (!(strncmp (" else", SubBuffer, 5)))
335         {
336             SubBuffer += 4;
337             NextBrace = strstr (SubBuffer, "{");
338             NextSemicolon = strstr (SubBuffer, ";");
339             NextIf = strstr (SubBuffer, " if");

341             if ((!NextBrace) ||
342                (NextSemicolon && (NextBrace > NextSemicolon)) ||
343                (NextIf && (NextBrace > NextIf)))
344             {
345                 Gbl_MissingBraces++;

347                 if (!Gbl_QuietMode)
348                 {
349                     printf ("Missing braces for <else>, line %u: %s\n", TotalLin
350                 }
351             }
352         }

354         SubBuffer++;
355     }
356 }


359 /******************************************************************************
360  *
361  * FUNCTION:    AsTrimLines
362  *
363  * DESCRIPTION: Remove extra blanks from the end of source lines. Does not
364  *              check for tabs.
365  *
366  *****************************************************************************/

368 void
369 AsTrimLines (
370     char                    *Buffer,
371     char                    *Filename)
372 {
373     char                    *SubBuffer = Buffer;
374     char                    *StartWhiteSpace = NULL;
375     UINT32                  SpaceCount = 0;


378     while (*SubBuffer)
379     {
380         while (*SubBuffer != '\n')
381         {
382             if (!*SubBuffer)
383             {
384                 goto Exit;
385             }

387             if (*SubBuffer == ' ')
388             {
389                 if (!StartWhiteSpace)
390                 {
```

```
391                     StartWhiteSpace = SubBuffer;
392                 }
393             }
394             else
395             {
396                 StartWhiteSpace = NULL;
397             }

399             SubBuffer++;
400         }

402         if (StartWhiteSpace)
403         {
404             SpaceCount += (SubBuffer - StartWhiteSpace);

406             /* Remove the spaces */

408             SubBuffer = AsRemoveData (StartWhiteSpace, SubBuffer);
409             StartWhiteSpace = NULL;
410         }

412         SubBuffer++;
413     }


416 Exit:
417     if (SpaceCount)
418     {
419         Gbl_MadeChanges = TRUE;
420         AsPrint ("Extraneous spaces removed", SpaceCount, Filename);
421     }
422 }


425 /******************************************************************************
426  *
427  * FUNCTION:    AsTrimWhitespace
428  *
429  * DESCRIPTION: Remove "excess" blank lines - any more than 2 blank lines.
430  *              this can happen during the translation when lines are removed.
431  *
432  *****************************************************************************/

434 void
435 AsTrimWhitespace (
436     char                    *Buffer)
437 {
438     int                     ReplaceCount = 1;


441     while (ReplaceCount)
442     {
443         ReplaceCount = AsReplaceString ("\n\n\n\n", "\n\n\n", REPLACE_SUBSTRINGS
444     }
445 }


448 /******************************************************************************
449  *
450  * FUNCTION:    AsReplaceHeader
451  *
452  * DESCRIPTION: Replace the default Intel legal header with a new header
453  *
454  *****************************************************************************/

456 void
```



```
457 AsReplaceHeader (
458     char                    *Buffer,
459     char                    *NewHeader)
460 {
461     char                    *SubBuffer;
462     char                    *TokenEnd;


465     /* Find the original header */

467     SubBuffer = strstr (Buffer, HeaderBegin);
468     if (!SubBuffer)
469     {
470         return;
471     }

473     /* Find the end of the original header */

475     TokenEnd = strstr (SubBuffer, "*/");
476     TokenEnd = AsSkipPastChar (TokenEnd, '\n');

478     /* Delete old header, insert new one */

480     AsReplaceData (SubBuffer, TokenEnd - SubBuffer, NewHeader, strlen (NewHeader
481 }


484 /******************************************************************************
485  *
486  * FUNCTION:    AsReplaceString
487  *
488  * DESCRIPTION: Replace all instances of a target string with a replacement
489  *              string. Returns count of the strings replaced.
490  *
491  *****************************************************************************/

493 int
494 AsReplaceString (
495     char                    *Target,
496     char                    *Replacement,
497     UINT8                   Type,
498     char                    *Buffer)
499 {
500     char                    *SubString1;
501     char                    *SubString2;
502     char                    *SubBuffer;
503     int                     TargetLength;
504     int                     ReplacementLength;
505     int                     ReplaceCount = 0;


508     TargetLength = strlen (Target);
509     ReplacementLength = strlen (Replacement);

511     SubBuffer = Buffer;
512     SubString1 = Buffer;

514     while (SubString1)
515     {
516         /* Find the target string */

518         SubString1 = strstr (SubBuffer, Target);
519         if (!SubString1)
520         {
521             return (ReplaceCount);
522         }
```

```
524         /*
525          * Check for translation escape string -- means to ignore
526          * blocks of code while replacing
527          */
528         if (Gbl_IgnoreTranslationEscapes)
529         {
530             SubString2 = NULL;
531         }
532         else
533         {
534             SubString2 = strstr (SubBuffer, AS_START_IGNORE);
535         }

537         if ((SubString2) &&
538             (SubString2 < SubString1))
539         {
540             /* Find end of the escape block starting at "Substring2" */

542             SubString2 = strstr (SubString2, AS_STOP_IGNORE);
543             if (!SubString2)
544             {
545                 /* Didn't find terminator */

547                 return (ReplaceCount);
548             }

550             /* Move buffer to end of escape block and continue */

552             SubBuffer = SubString2;
553         }

555         /* Do the actual replace if the target was found */

557         else
558         {
559             if ((Type & REPLACE_MASK) == REPLACE_WHOLE_WORD)
560             {
561                 if (!AsMatchExactWord (SubString1, TargetLength))
562                 {
563                     SubBuffer = SubString1 + 1;
564                     continue;
565                 }
566             }

568             SubBuffer = AsReplaceData (SubString1, TargetLength, Replacement, Re

570             if ((Type & EXTRA_INDENT_C) &&
571                 (!Gbl_StructDefs))
572             {
573                 SubBuffer = AsInsertData (SubBuffer, "        ", 8);
574             }

576             ReplaceCount++;
577         }
578     }

580     return (ReplaceCount);
581 }


584 /*******************************************************************************
585  *
586  * FUNCTION:    AsConvertToLineFeeds
587  *
588  * DESCRIPTION:
```



```
589  *
590  ******************************************************************************/

592 void
593 AsConvertToLineFeeds (
594     char                    *Buffer)
595 {
596     char                    *SubString;
597     char                    *SubBuffer;


600     SubBuffer = Buffer;
601     SubString = Buffer;

603     while (SubString)
604     {
605         /* Find the target string */

607         SubString = strstr (SubBuffer, "\r\n");
608         if (!SubString)
609         {
610             return;
611         }

613         SubBuffer = AsReplaceData (SubString, 1, NULL, 0);
614     }
615     return;
616 }


619 /*******************************************************************************
620  *
621  * FUNCTION:    AsInsertCarriageReturns
622  *
623  * DESCRIPTION:
624  *
625  ******************************************************************************/

627 void
628 AsInsertCarriageReturns (
629     char                    *Buffer)
630 {
631     char                    *SubString;
632     char                    *SubBuffer;


635     SubBuffer = Buffer;
636     SubString = Buffer;

638     while (SubString)
639     {
640         /* Find the target string */

642         SubString = strstr (SubBuffer, "\n");
643         if (!SubString)
644         {
645             return;
646         }

648         SubBuffer = AsInsertData (SubString, "\r", 1);
649         SubBuffer += 1;
650     }
651     return;
652 }
```

```
/*****************************************************************************
 *
 * FUNCTION:    AsBracesOnSameLine
 *
 * DESCRIPTION: Move opening braces up to the same line as an if, for, else,
 *              or while statement (leave function opening brace on separate
 *              line).
 *
 ****************************************************************************/

void
AsBracesOnSameLine (
    char                    *Buffer)
{
    char                    *SubBuffer = Buffer;
    char                    *Beginning;
    char                    *StartOfThisLine;
    char                    *Next;
    BOOLEAN                 BlockBegin = TRUE;


    while (*SubBuffer)
    {
        /* Ignore comments */

        if ((SubBuffer[0] == '/') &&
            (SubBuffer[1] == '*'))
        {
            SubBuffer = strstr (SubBuffer, "*/");
            if (!SubBuffer)
            {
                return;
            }

            SubBuffer += 2;
            continue;
        }

        /* Ignore quoted strings */

        if (*SubBuffer == '\"')
        {
            SubBuffer++;
            SubBuffer = AsSkipPastChar (SubBuffer, '\"');
            if (!SubBuffer)
            {
                return;
            }
        }

        if (!strncmp ("\n}", SubBuffer, 2))
        {
            /*
             * A newline followed by a closing brace closes a function
             * or struct or initializer block
             */
            BlockBegin = TRUE;
        }

        /*
         * Move every standalone brace up to the previous line
         * Check for digit will ignore initializer lists surrounded by braces.
         * This will work until we we need more complex detection.
         */
        if ((*SubBuffer == '{') && !isdigit ((int) SubBuffer[1]))
        {
```



```
            if (BlockBegin)
            {
                BlockBegin = FALSE;
            }
            else
            {
                /*
                 * Backup to previous non-whitespace
                 */
                Beginning = SubBuffer - 1;
                while ((*Beginning == ' ')   ||
                       (*Beginning == '\n'))
                {
                    Beginning--;
                }

                StartOfThisLine = Beginning;
                while (*StartOfThisLine != '\n')
                {
                    StartOfThisLine--;
                }

                /*
                 * Move the brace up to the previous line, UNLESS:
                 *
                 * 1) There is a conditional compile on the line (starts with '#
                 * 2) Previous line ends with an '=' (Start of initializer block
                 * 3) Previous line ends with a comma (part of an init list)
                 * 4) Previous line ends with a backslash (part of a macro)
                 */
                if ((StartOfThisLine[1] != '#') &&
                    (*Beginning != '\\') &&
                    (*Beginning != '/') &&
                    (*Beginning != '{') &&
                    (*Beginning != '=') &&
                    (*Beginning != ','))
                {
                    Beginning++;
                    SubBuffer++;

                    Gbl_MadeChanges = TRUE;

#ifdef ADD_EXTRA_WHITESPACE
                    AsReplaceData (Beginning, SubBuffer - Beginning, " {\n", 3);
#else
                    /* Find non-whitespace start of next line */

                    Next = SubBuffer + 1;
                    while ((*Next == ' ')   ||
                           (*Next == '\t'))
                    {
                        Next++;
                    }

                    /* Find non-whitespace start of this line */

                    StartOfThisLine++;
                    while ((*StartOfThisLine == ' ')   ||
                           (*StartOfThisLine == '\t'))
                    {
                        StartOfThisLine++;
                    }

                    /*
                     * Must be a single-line comment to need more whitespace
                     * Even then, we don't need more if the previous statement
```

```
787                      * is an "else".
788                      */
789                     if ((Next[0] == '/')  &&
790                         (Next[1] == '*')  &&
791                         (Next[2] != '\n') &&
792
793                         (!strncmp (StartOfThisLine, "else if", 7)     ||
794                          !strncmp (StartOfThisLine, "else while", 10) ||
795                           strncmp (StartOfThisLine, "else", 4)))
796                     {
797                         AsReplaceData (Beginning, SubBuffer - Beginning, " {\n",
798                     }
799                     else
800                     {
801                         AsReplaceData (Beginning, SubBuffer - Beginning, " {", 2
802                     }
803 #endif
804                 }
805             }
806         }
807
808         SubBuffer++;
809     }
810 }
811
812
813 /******************************************************************************
814  *
815  * FUNCTION:    AsTabify4
816  *
817  * DESCRIPTION: Convert the text to tabbed text. Alignment of text is
818  *              preserved.
819  *
820  *****************************************************************************/
821
822 void
823 AsTabify4 (
824     char                    *Buffer)
825 {
826     char                    *SubBuffer = Buffer;
827     char                    *NewSubBuffer;
828     UINT32                  SpaceCount = 0;
829     UINT32                  Column = 0;
830
831
832     while (*SubBuffer)
833     {
834         if (*SubBuffer == '\n')
835         {
836             Column = 0;
837         }
838         else
839         {
840             Column++;
841         }
842
843         /* Ignore comments */
844
845         if ((SubBuffer[0] == '/') &&
846             (SubBuffer[1] == '*'))
847         {
848             SubBuffer = strstr (SubBuffer, "*/");
849             if (!SubBuffer)
850             {
851                 return;
852             }
```



```
853
854             SubBuffer += 2;
855             continue;
856         }
857
858         /* Ignore quoted strings */
859
860         if (*SubBuffer == '\"')
861         {
862             SubBuffer++;
863             SubBuffer = AsSkipPastChar (SubBuffer, '\"');
864             if (!SubBuffer)
865             {
866                 return;
867             }
868             SpaceCount = 0;
869         }
870
871         if (*SubBuffer == ' ')
872         {
873             SpaceCount++;
874
875             if (SpaceCount >= 4)
876             {
877                 SpaceCount = 0;
878
879                 NewSubBuffer = (SubBuffer + 1) - 4;
880                 *NewSubBuffer = '\t';
881                 NewSubBuffer++;
882
883                 /* Remove the spaces */
884
885                 SubBuffer = AsRemoveData (NewSubBuffer, SubBuffer + 1);
886             }
887
888             if ((Column % 4) == 0)
889             {
890                 SpaceCount = 0;
891             }
892         }
893         else
894         {
895             SpaceCount = 0;
896         }
897
898         SubBuffer++;
899     }
900 }
901
902
903 /******************************************************************************
904  *
905  * FUNCTION:    AsTabify8
906  *
907  * DESCRIPTION: Convert the text to tabbed text. Alignment of text is
908  *              preserved.
909  *
910  *****************************************************************************/
911
912 void
913 AsTabify8 (
914     char                    *Buffer)
915 {
916     char                    *SubBuffer = Buffer;
917     char                    *NewSubBuffer;
918     char                    *CommentEnd = NULL;
```

```
    UINT32                  SpaceCount = 0;
    UINT32                  Column = 0;
    UINT32                  TabCount = 0;
    UINT32                  LastLineTabCount = 0;
    UINT32                  LastLineColumnStart = 0;
    UINT32                  ThisColumnStart = 0;
    UINT32                  ThisTabCount =  0;
    char                    *FirstNonBlank = NULL;


    while (*SubBuffer)
    {
        if (*SubBuffer == '\n')
        {
            /* This is a standalone blank line */

            FirstNonBlank = NULL;
            Column = 0;
            SpaceCount = 0;
            TabCount = 0;
            SubBuffer++;
            continue;
        }

        if (!FirstNonBlank)
        {
            /* Find the first non-blank character on this line */

            FirstNonBlank = SubBuffer;
            while (*FirstNonBlank == ' ')
            {
                FirstNonBlank++;
            }

            /*
             * This mechanism limits the difference in tab counts from
             * line to line. It helps avoid the situation where a second
             * continuation line (which was indented correctly for tabs=4) would
             * get indented off the screen if we just blindly converted to tabs.
             */
            ThisColumnStart = FirstNonBlank - SubBuffer;

            if (LastLineTabCount == 0)
            {
                ThisTabCount = 0;
            }
            else if (ThisColumnStart == LastLineColumnStart)
            {
                ThisTabCount = LastLineTabCount -1;
            }
            else
            {
                ThisTabCount = LastLineTabCount + 1;
            }
        }

        Column++;

        /* Check if we are in a comment */

        if ((SubBuffer[0] == '*') &&
            (SubBuffer[1] == '/'))
        {
            SpaceCount = 0;
            SubBuffer += 2;
```



```
            if (*SubBuffer == '\n')
            {
                if (TabCount > 0)
                {
                    LastLineTabCount = TabCount;
                    TabCount = 0;
                }
                FirstNonBlank = NULL;
                LastLineColumnStart = ThisColumnStart;
                SubBuffer++;
            }

            continue;
        }

        /* Check for comment open */

        if ((SubBuffer[0] == '/') &&
            (SubBuffer[1] == '*'))
        {
            /* Find the end of the comment, it must exist */

            CommentEnd = strstr (SubBuffer, "*/");
            if (!CommentEnd)
            {
                return;
            }

            /* Toss the rest of this line or single-line comment */

            while ((SubBuffer < CommentEnd) &&
                   (*SubBuffer != '\n'))
            {
                SubBuffer++;
            }

            if (*SubBuffer == '\n')
            {
                if (TabCount > 0)
                {
                    LastLineTabCount = TabCount;
                    TabCount = 0;
                }
                FirstNonBlank = NULL;
                LastLineColumnStart = ThisColumnStart;
            }

            SpaceCount = 0;
            continue;
        }

        /* Ignore quoted strings */

        if ((!CommentEnd) && (*SubBuffer == '\"'))
        {
            SubBuffer++;
            SubBuffer = AsSkipPastChar (SubBuffer, '\"');
            if (!SubBuffer)
            {
                return;
            }
            SpaceCount = 0;
        }

        if (*SubBuffer != ' ')
        {
```

```
            /* Not a space, skip to end of line */

            SubBuffer = AsSkipUntilChar (SubBuffer, '\n');
            if (!SubBuffer)
            {
                return;
            }
            if (TabCount > 0)
            {
                LastLineTabCount = TabCount;
                TabCount = 0;
            }

            FirstNonBlank = NULL;
            LastLineColumnStart = ThisColumnStart;
            Column = 0;
            SpaceCount = 0;
        }
        else
        {
            /* Another space */

            SpaceCount++;

            if (SpaceCount >= 4)
            {
                /* Replace this group of spaces with a tab character */

                SpaceCount = 0;

                NewSubBuffer = SubBuffer - 3;

                if (TabCount <= ThisTabCount ? (ThisTabCount +1) : 0)
                {
                    *NewSubBuffer = '\t';
                    NewSubBuffer++;
                    SubBuffer++;
                    TabCount++;
                }

                /* Remove the spaces */

                SubBuffer = AsRemoveData (NewSubBuffer, SubBuffer);
                continue;
            }
        }

        SubBuffer++;
    }
}


/******************************************************************************
 *
 * FUNCTION:    AsCountLines
 *
 * DESCRIPTION: Count the number of lines in the input buffer. Also count
 *              the number of long lines (lines longer than 80 chars).
 *
 *****************************************************************************/

UINT32
AsCountLines (
    char                    *Buffer,
    char                    *Filename)
{
```



```
    char                    *SubBuffer = Buffer;
    char                    *EndOfLine;
    UINT32                  LineCount = 0;
    UINT32                  LongLineCount = 0;


    while (*SubBuffer)
    {
        EndOfLine = AsSkipUntilChar (SubBuffer, '\n');
        if (!EndOfLine)
        {
            Gbl_TotalLines += LineCount;
            return (LineCount);
        }

        if ((EndOfLine - SubBuffer) > 80)
        {
            LongLineCount++;
            VERBOSE_PRINT (("long: %.80s\n", SubBuffer));
        }

        LineCount++;
        SubBuffer = EndOfLine + 1;
    }

    if (LongLineCount)
    {
        VERBOSE_PRINT (("%u Lines longer than 80 found in %s\n", LongLineCount,
        Gbl_LongLines += LongLineCount;
    }

    Gbl_TotalLines += LineCount;
    return (LineCount);
}


/******************************************************************************
 *
 * FUNCTION:    AsCountTabs
 *
 * DESCRIPTION: Simply count the number of tabs in the input file buffer
 *
 *****************************************************************************/

void
AsCountTabs (
    char                    *Buffer,
    char                    *Filename)
{
    UINT32                  i;
    UINT32                  TabCount = 0;


    for (i = 0; Buffer[i]; i++)
    {
        if (Buffer[i] == '\t')
        {
            TabCount++;
        }
    }

    if (TabCount)
    {
        AsPrint ("Tabs found", TabCount, Filename);
        Gbl_Tabs += TabCount;
    }
```

```
1184     AsCountLines (Buffer, Filename);
1185 }


1188 /******************************************************************************
1189  *
1190  * FUNCTION:    AsCountNonAnsiComments
1191  *
1192  * DESCRIPTION: Count the number of "//" comments. This type of comment is
1193  *              non-ANSI C.
1194  *
1195  *****************************************************************************/

1197 void
1198 AsCountNonAnsiComments (
1199     char                    *Buffer,
1200     char                    *Filename)
1201 {
1202     char                    *SubBuffer = Buffer;
1203     UINT32                  CommentCount = 0;


1206     while (SubBuffer)
1207     {
1208         SubBuffer = strstr (SubBuffer, "//");
1209         if (SubBuffer)
1210         {
1211             CommentCount++;
1212             SubBuffer += 2;
1213         }
1214     }

1216     if (CommentCount)
1217     {
1218         AsPrint ("Non-ANSI Comments found", CommentCount, Filename);
1219         Gbl_NonAnsiComments += CommentCount;
1220     }
1221 }


1224 /******************************************************************************
1225  *
1226  * FUNCTION:    AsCountSourceLines
1227  *
1228  * DESCRIPTION: Count the number of C source lines. Defined by 1) not a
1229  *              comment, and 2) not a blank line.
1230  *
1231  *****************************************************************************/

1233 void
1234 AsCountSourceLines (
1235     char                    *Buffer,
1236     char                    *Filename)
1237 {
1238     char                    *SubBuffer = Buffer;
1239     UINT32                  LineCount = 0;
1240     UINT32                  WhiteCount = 0;
1241     UINT32                  CommentCount = 0;


1244     while (*SubBuffer)
1245     {
1246         /* Detect comments (// comments are not used, non-ansii) */

1248         if ((SubBuffer[0] == '/') &&
```



```
1249             (SubBuffer[1] == '*'))
1250         {
1251             SubBuffer += 2;

1253             /* First line of multi-line comment is often just whitespace */

1255             if (SubBuffer[0] == '\n')
1256             {
1257                 WhiteCount++;
1258                 SubBuffer++;
1259             }
1260             else
1261             {
1262                 CommentCount++;
1263             }

1265             /* Find end of comment */

1267             while (SubBuffer[0] && SubBuffer[1] &&
1268                    !(((SubBuffer[0] == '*') &&
1269                      (SubBuffer[1] == '/'))))
1270             {
1271                 if (SubBuffer[0] == '\n')
1272                 {
1273                     CommentCount++;
1274                 }

1276                 SubBuffer++;
1277             }
1278         }

1280         /* A linefeed followed by a non-linefeed is a valid source line */

1282         else if ((SubBuffer[0] == '\n') &&
1283                  (SubBuffer[1] != '\n'))
1284         {
1285             LineCount++;
1286         }

1288         /* Two back-to-back linefeeds indicate a whitespace line */

1290         else if ((SubBuffer[0] == '\n') &&
1291                  (SubBuffer[1] == '\n'))
1292         {
1293             WhiteCount++;
1294         }

1296         SubBuffer++;
1297     }

1299     /* Adjust comment count for legal header */

1301     if (Gbl_HeaderSize < CommentCount)
1302     {
1303         CommentCount -= Gbl_HeaderSize;
1304         Gbl_HeaderLines += Gbl_HeaderSize;
1305     }

1307     Gbl_SourceLines += LineCount;
1308     Gbl_WhiteLines += WhiteCount;
1309     Gbl_CommentLines += CommentCount;

1311     VERBOSE_PRINT (("%u Comment %u White %u Code %u Lines in %s\n",
1312                     CommentCount, WhiteCount, LineCount, LineCount+WhiteCount+Commen
1313 }
```

```
1316 /*****************************************************************************
1317  *
1318  * FUNCTION:    AsInsertPrefix
1319  *
1320  * DESCRIPTION: Insert struct or union prefixes
1321  *
1322  ****************************************************************************/

1324 void
1325 AsInsertPrefix (
1326     char                    *Buffer,
1327     char                    *Keyword,
1328     UINT8                   Type)
1329 {
1330     char                    *SubString;
1331     char                    *SubBuffer;
1332     char                    *EndKeyword;
1333     int                     InsertLength;
1334     char                    *InsertString;
1335     int                     TrailingSpaces;
1336     char                    LowerKeyword[128];
1337     int                     KeywordLength;


1340     switch (Type)
1341     {
1342     case SRC_TYPE_STRUCT:

1344         InsertString = "struct ";
1345         break;

1347     case SRC_TYPE_UNION:

1349         InsertString = "union ";
1350         break;

1352     default:

1354         return;
1355     }

1357     strcpy (LowerKeyword, Keyword);
1358     AsStrlwr (LowerKeyword);

1360     SubBuffer = Buffer;
1361     SubString = Buffer;
1362     InsertLength = strlen (InsertString);
1363     KeywordLength = strlen (Keyword);


1366     while (SubString)
1367     {
1368         /* Find an instance of the keyword */

1370         SubString = strstr (SubBuffer, LowerKeyword);
1371         if (!SubString)
1372         {
1373             return;
1374         }

1376         SubBuffer = SubString;

1378         /* Must be standalone word, not a substring */

1380         if (AsMatchExactWord (SubString, KeywordLength))
```



```
1381         {
1382             /* Make sure the keyword isn't already prefixed with the insert */

1384             if (!strncmp (SubString - InsertLength, InsertString, InsertLength))
1385             {
1386                 /* Add spaces if not already at the end-of-line */

1388                 if (*(SubBuffer + KeywordLength) != '\n')
1389                 {
1390                     /* Already present, add spaces after to align structure memb

1392 #if 0
1393 /* ONLY FOR C FILES */
1394                     AsInsertData (SubBuffer + KeywordLength, "        ", 8);
1395 #endif
1396                 }
1397                 goto Next;
1398             }

1400             /* Make sure the keyword isn't at the end of a struct/union */
1401             /* Note: This code depends on a single space after the brace */

1403             if (*(SubString - 2) == '}')
1404             {
1405                 goto Next;
1406             }

1408             /* Prefix the keyword with the insert string */

1410             Gbl_MadeChanges = TRUE;

1412             /* Is there room for insertion */

1414             EndKeyword = SubString + strlen (LowerKeyword);

1416             TrailingSpaces = 0;
1417             while (EndKeyword[TrailingSpaces] == ' ')
1418             {
1419                 TrailingSpaces++;
1420             }

1422             /*
1423              * Use "if (TrailingSpaces > 1)" if we want to ignore casts
1424              */
1425             SubBuffer = SubString + InsertLength;

1427             if (TrailingSpaces > InsertLength)
1428             {
1429                 /* Insert the keyword */

1431                 memmove (SubBuffer, SubString, KeywordLength);

1433                 /* Insert the keyword */

1435                 memmove (SubString, InsertString, InsertLength);
1436             }
1437             else
1438             {
1439                 AsInsertData (SubString, InsertString, InsertLength);
1440             }
1441         }

1443 Next:
1444         SubBuffer += KeywordLength;
1445     }
1446 }
```

```
#ifdef ACPI_FUTURE_IMPLEMENTATION
/******************************************************************************
 *
 * FUNCTION:    AsTrimComments
 *
 * DESCRIPTION: Finds 3-line comments with only a single line of text
 *
 *****************************************************************************/

void
AsTrimComments (
    char                    *Buffer,
    char                    *Filename)
{
    char                    *SubBuffer = Buffer;
    char                    *Ptr1;
    char                    *Ptr2;
    UINT32                  LineCount;
    UINT32                  ShortCommentCount = 0;


    while (1)
    {
        /* Find comment open, within procedure level */

        SubBuffer = strstr (SubBuffer, "    /*");
        if (!SubBuffer)
        {
            goto Exit;
        }

        /* Find comment terminator */

        Ptr1 = strstr (SubBuffer, "*/");
        if (!Ptr1)
        {
            goto Exit;
        }

        /* Find next EOL (from original buffer) */

        Ptr2 = strstr (SubBuffer, "\n");
        if (!Ptr2)
        {
            goto Exit;
        }

        /* Ignore one-line comments */

        if (Ptr1 < Ptr2)
        {
            /* Normal comment, ignore and continue; */

            SubBuffer = Ptr2;
            continue;
        }

        /* Examine multi-line comment */

        LineCount = 1;
        while (Ptr1 > Ptr2)
        {
            /* Find next EOL */

            Ptr2++;
```



```
            Ptr2 = strstr (Ptr2, "\n");
            if (!Ptr2)
            {
                goto Exit;
            }

            LineCount++;
        }

        SubBuffer = Ptr1;

        if (LineCount <= 3)
        {
            ShortCommentCount++;
        }
    }


Exit:

    if (ShortCommentCount)
    {
        AsPrint ("Short Comments found", ShortCommentCount, Filename);
    }
}
#endif
```

```
**********************************************************
   22379 Thu Dec 26 13:50:12 2013
new/usr/src/common/acpica/tools/acpisrc/asfile.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asfile - Main module for the acpi source processor utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpisrc.h"

/* Local prototypes */

void
AsDoWildcard (
    ACPI_CONVERSION_TABLE   *ConversionTable,
    char                    *SourcePath,
    char                    *TargetPath,
    int                     MaxPathLength,
    int                     FileType,
    char                    *WildcardSpec);

BOOLEAN
AsDetectLoneLineFeeds (
    char                    *Filename,
    char                    *Buffer);
```



```
static ACPI_INLINE int
AsMaxInt (int a, int b)
{
    return (a > b ? a : b);
}


/******************************************************************************
 *
 * FUNCTION:    AsDoWildcard
 *
 * DESCRIPTION: Process files via wildcards
 *
 *****************************************************************************/

void
AsDoWildcard (
    ACPI_CONVERSION_TABLE   *ConversionTable,
    char                    *SourcePath,
    char                    *TargetPath,
    int                     MaxPathLength,
    int                     FileType,
    char                    *WildcardSpec)
{
    void                    *DirInfo;
    char                    *Filename;
    char                    *SourceDirPath;
    char                    *TargetDirPath;
    char                    RequestedFileType;


    if (FileType == FILE_TYPE_DIRECTORY)
    {
        RequestedFileType = REQUEST_DIR_ONLY;
    }
    else
    {
        RequestedFileType = REQUEST_FILE_ONLY;
    }

    VERBOSE_PRINT (("Checking for %s source files in directory \"%s\"\n",
            WildcardSpec, SourcePath));

    /* Open the directory for wildcard search */

    DirInfo = AcpiOsOpenDirectory (SourcePath, WildcardSpec, RequestedFileType);
    if (DirInfo)
    {
        /*
         * Get all of the files that match both the
         * wildcard and the requested file type
         */
        while ((Filename = AcpiOsGetNextFilename (DirInfo)))
        {
            /* Looking for directory files, must check file type */

            switch (RequestedFileType)
            {
            case REQUEST_DIR_ONLY:

                /* If we actually have a dir, process the subtree */

                if (!AsCheckForDirectory (SourcePath, TargetPath, Filename,
                        &SourceDirPath, &TargetDirPath))
                {
```

```
127                     VERBOSE_PRINT (("Subdirectory: %s\n", Filename));
129                     AsProcessTree (ConversionTable, SourceDirPath, TargetDirPath
130                     free (SourceDirPath);
131                     free (TargetDirPath);
132                 }
133                 break;
135             case REQUEST_FILE_ONLY:
137                 /* Otherwise, this is a file, not a directory */
139                 VERBOSE_PRINT (("File: %s\n", Filename));
141                 AsProcessOneFile (ConversionTable, SourcePath, TargetPath,
142                         MaxPathLength, Filename, FileType);
143                 break;
145             default:
147                 break;
148             }
149         }
151         /* Cleanup */
153         AcpiOsCloseDirectory (DirInfo);
154     }
155 }


158 /******************************************************************************
159  *
160  * FUNCTION:    AsProcessTree
161  *
162  * DESCRIPTION: Process the directory tree. Files with the extension ".C" and
163  *              ".H" are processed as the tree is traversed.
164  *
165  *****************************************************************************/

167 ACPI_NATIVE_INT
168 AsProcessTree (
169     ACPI_CONVERSION_TABLE   *ConversionTable,
170     char                    *SourcePath,
171     char                    *TargetPath)
172 {
173     int                     MaxPathLength;


176     MaxPathLength = AsMaxInt (strlen (SourcePath), strlen (TargetPath));

178     if (!(ConversionTable->Flags & FLG_NO_FILE_OUTPUT))
179     {
180         if (ConversionTable->Flags & FLG_LOWERCASE_DIRNAMES)
181         {
182             AsStrlwr (TargetPath);
183         }

185         VERBOSE_PRINT (("Creating Directory \"%s\"\n", TargetPath));
186         if (mkdir (TargetPath))
187         {
188             if (errno != EEXIST)
189             {
190                 printf ("Could not create target directory\n");
191                 return (-1);
192             }
```



```
193         }
194     }

196     /* Do the C source files */

198     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
199             FILE_TYPE_SOURCE, "*.c");

201     /* Do the C header files */

203     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
204             FILE_TYPE_HEADER, "*.h");

206     /* Do the Lex file(s) */

208     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
209             FILE_TYPE_SOURCE, "*.l");

211     /* Do the yacc file(s) */

213     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
214             FILE_TYPE_SOURCE, "*.y");

216     /* Do any ASL files */

218     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
219             FILE_TYPE_HEADER, "*.asl");

221     /* Do any subdirectories */

223     AsDoWildcard (ConversionTable, SourcePath, TargetPath, MaxPathLength,
224             FILE_TYPE_DIRECTORY, "*");

226     return (0);
227 }


230 /******************************************************************************
231  *
232  * FUNCTION:    AsDetectLoneLineFeeds
233  *
234  * DESCRIPTION: Find LF without CR.
235  *
236  *****************************************************************************/

238 BOOLEAN
239 AsDetectLoneLineFeeds (
240     char                    *Filename,
241     char                    *Buffer)
242 {
243     UINT32                  i = 1;
244     UINT32                  LfCount = 0;
245     UINT32                  LineCount = 0;


248     if (!Buffer[0])
249     {
250         return (FALSE);
251     }

253     while (Buffer[i])
254     {
255         if (Buffer[i] == 0x0A)
256         {
257             if (Buffer[i-1] != 0x0D)
258             {
```

```
259                 LfCount++;
260             }
261             LineCount++;
262         }
263         i++;
264     }

266     if (LfCount)
267     {
268         if (LineCount == LfCount)
269         {
270             if (!Gbl_IgnoreLoneLineFeeds)
271             {
272                 printf ("%s: ****File has UNIX format**** (LF only, not CR/LF) %
273                     Filename, LfCount);
274             }
275         }
276         else
277         {
278             printf ("%s: %u lone linefeeds in file\n", Filename, LfCount);
279         }
280         return (TRUE);
281     }

283     return (FALSE);
284 }


287 /******************************************************************************
288  *
289  * FUNCTION:    AsConvertFile
290  *
291  * DESCRIPTION: Perform the requested transforms on the file buffer (as
292  *              determined by the ConversionTable and the FileType).
293  *
294  *****************************************************************************/

296 void
297 AsConvertFile (
298     ACPI_CONVERSION_TABLE   *ConversionTable,
299     char                    *FileBuffer,
300     char                    *Filename,
301     ACPI_NATIVE_INT         FileType)
302 {
303     UINT32                  i;
304     UINT32                  Functions;
305     ACPI_STRING_TABLE       *StringTable;
306     ACPI_IDENTIFIER_TABLE   *ConditionalTable;
307     ACPI_IDENTIFIER_TABLE   *LineTable;
308     ACPI_IDENTIFIER_TABLE   *MacroTable;
309     ACPI_TYPED_IDENTIFIER_TABLE *StructTable;
310     ACPI_IDENTIFIER_TABLE   *SpecialMacroTable;


313     switch (FileType)
314     {
315     case FILE_TYPE_SOURCE:

317         Functions           = ConversionTable->SourceFunctions;
318         StringTable         = ConversionTable->SourceStringTable;
319         LineTable           = ConversionTable->SourceLineTable;
320         ConditionalTable    = ConversionTable->SourceConditionalTable;
321         MacroTable          = ConversionTable->SourceMacroTable;
322         StructTable         = ConversionTable->SourceStructTable;
323         SpecialMacroTable   = ConversionTable->SourceSpecialMacroTable;
324        break;
```



```
326     case FILE_TYPE_HEADER:

328         Functions           = ConversionTable->HeaderFunctions;
329         StringTable         = ConversionTable->HeaderStringTable;
330         LineTable           = ConversionTable->HeaderLineTable;
331         ConditionalTable    = ConversionTable->HeaderConditionalTable;
332         MacroTable          = ConversionTable->HeaderMacroTable;
333         StructTable         = ConversionTable->HeaderStructTable;
334         SpecialMacroTable   = ConversionTable->HeaderSpecialMacroTable;
335         break;

337     default:

339         printf ("Unknown file type, cannot process\n");
340         return;
341     }


344     Gbl_StructDefs = strstr (FileBuffer, "/* acpisrc:StructDefs");
345     Gbl_Files++;
346     VERBOSE_PRINT (("Processing %u bytes\n",
347         (unsigned int) strlen (FileBuffer)));

349     if (Gbl_Cleanup)
350     {
351         AsRemoveExtraLines (FileBuffer, Filename);
352         AsRemoveSpacesAfterPeriod (FileBuffer, Filename);
353     }

355     if (ConversionTable->LowerCaseTable)
356     {
357         for (i = 0; ConversionTable->LowerCaseTable[i].Identifier; i++)
358         {
359             AsLowerCaseString (ConversionTable->LowerCaseTable[i].Identifier,
360                                FileBuffer);
361         }
362     }

364     /* Process all the string replacements */

366     if (StringTable)
367     {
368         for (i = 0; StringTable[i].Target; i++)
369         {
370             AsReplaceString (StringTable[i].Target, StringTable[i].Replacement,
371                     StringTable[i].Type, FileBuffer);
372         }
373     }

375     if (LineTable)
376     {
377         for (i = 0; LineTable[i].Identifier; i++)
378         {
379             AsRemoveLine (FileBuffer, LineTable[i].Identifier);
380         }
381     }

383     if (ConditionalTable)
384     {
385         for (i = 0; ConditionalTable[i].Identifier; i++)
386         {
387             AsRemoveConditionalCompile (FileBuffer, ConditionalTable[i].Identifi
388         }
389     }
```

```
391     if (MacroTable)
392     {
393         for (i = 0; MacroTable[i].Identifier; i++)
394         {
395             AsRemoveMacro (FileBuffer, MacroTable[i].Identifier);
396         }
397     }

399     if (StructTable)
400     {
401         for (i = 0; StructTable[i].Identifier; i++)
402         {
403             AsInsertPrefix (FileBuffer, StructTable[i].Identifier, StructTable[i
404         }
405     }

407     if (SpecialMacroTable)
408     {
409         for (i = 0; SpecialMacroTable[i].Identifier; i++)
410         {
411             AsCleanupSpecialMacro (FileBuffer, SpecialMacroTable[i].Identifier);
412         }
413     }

415     /* Process the function table */

417     for (i = 0; i < 32; i++)
418     {
419         /* Decode the function bitmap */

421         switch ((1 << i) & Functions)
422         {
423         case 0:

425             /* This function not configured */
426             break;

428         case CVT_COUNT_TABS:

430             AsCountTabs (FileBuffer, Filename);
431             break;

433         case CVT_COUNT_NON_ANSI_COMMENTS:

435             AsCountNonAnsiComments (FileBuffer, Filename);
436             break;

438         case CVT_CHECK_BRACES:

440             AsCheckForBraces (FileBuffer, Filename);
441             break;

443         case CVT_TRIM_LINES:

445             AsTrimLines (FileBuffer, Filename);
446             break;

448         case CVT_COUNT_LINES:

450             AsCountSourceLines (FileBuffer, Filename);
451             break;

453         case CVT_BRACES_ON_SAME_LINE:

455             AsBracesOnSameLine (FileBuffer);
456             break;
```



```
458         case CVT_MIXED_CASE_TO_UNDERSCORES:

460             AsMixedCaseToUnderscores (FileBuffer, Filename);
461             break;

463         case CVT_LOWER_CASE_IDENTIFIERS:

465             AsLowerCaseIdentifiers (FileBuffer);
466             break;

468         case CVT_REMOVE_DEBUG_MACROS:

470             AsRemoveDebugMacros (FileBuffer);
471             break;

473         case CVT_TRIM_WHITESPACE:

475             AsTrimWhitespace (FileBuffer);
476             break;

478         case CVT_REMOVE_EMPTY_BLOCKS:

480             AsRemoveEmptyBlocks (FileBuffer, Filename);
481             break;

483         case CVT_REDUCE_TYPEDEFS:

485             AsReduceTypedefs (FileBuffer, "typedef union");
486             AsReduceTypedefs (FileBuffer, "typedef struct");
487             break;

489         case CVT_SPACES_TO_TABS4:

491             AsTabify4 (FileBuffer);
492             break;

494         case CVT_SPACES_TO_TABS8:

496             AsTabify8 (FileBuffer);
497             break;

499         case CVT_COUNT_SHORTMULTILINE_COMMENTS:

501 #ifdef ACPI_FUTURE_IMPLEMENTATION
502             AsTrimComments (FileBuffer, Filename);
503 #endif
504             break;

506         default:

508             printf ("Unknown conversion subfunction opcode\n");
509             break;
510         }
511     }

513     if (ConversionTable->NewHeader)
514     {
515         AsReplaceHeader (FileBuffer, ConversionTable->NewHeader);
516     }
517 }


520 /******************************************************************************
521  *
522  * FUNCTION:    AsProcessOneFile
```

```
 *
 * DESCRIPTION: Process one source file. The file is opened, read entirely
 *              into a buffer, converted, then written to a new file.
 *
 *****************************************************************************/

ACPI_NATIVE_INT
AsProcessOneFile (
    ACPI_CONVERSION_TABLE   *ConversionTable,
    char                    *SourcePath,
    char                    *TargetPath,
    int                     MaxPathLength,
    char                    *Filename,
    ACPI_NATIVE_INT         FileType)
{
    char                    *Pathname;
    char                    *OutPathname = NULL;


    /* Allocate a file pathname buffer for both source and target */

    Pathname = calloc (MaxPathLength + strlen (Filename) + 2, 1);
    if (!Pathname)
    {
        printf ("Could not allocate buffer for file pathnames\n");
        return (-1);
    }

    Gbl_FileType = FileType;

    /* Generate the source pathname and read the file */

    if (SourcePath)
    {
        strcpy (Pathname, SourcePath);
        strcat (Pathname, "/");
    }

    strcat (Pathname, Filename);

    if (AsGetFile (Pathname, &Gbl_FileBuffer, &Gbl_FileSize))
    {
        return (-1);
    }

    Gbl_HeaderSize = 0;
    if (strstr (Filename, ".asl"))
    {
        Gbl_HeaderSize = LINES_IN_ASL_HEADER; /* Lines in default ASL header */
    }
    else if (strstr (Gbl_FileBuffer, LEGAL_HEADER_SIGNATURE))
    {
        Gbl_HeaderSize = LINES_IN_LEGAL_HEADER; /* Normal C file and H header */
    }
    else if (strstr (Gbl_FileBuffer, LINUX_HEADER_SIGNATURE))
    {
        Gbl_HeaderSize = LINES_IN_LINUX_HEADER; /* Linuxized C file and H header
    }

    /* Process the file in the buffer */

    Gbl_MadeChanges = FALSE;
    if (!Gbl_IgnoreLoneLineFeeds && Gbl_HasLoneLineFeeds)
    {
        /*
         * All lone LFs will be converted to CR/LF
```



```
         * (when file is written, Windows version only)
         */
        printf ("Converting lone linefeeds\n");
        Gbl_MadeChanges = TRUE;
    }

    AsConvertFile (ConversionTable, Gbl_FileBuffer, Pathname, FileType);

    if (!(ConversionTable->Flags & FLG_NO_FILE_OUTPUT))
    {
        if (!(Gbl_Overwrite && !Gbl_MadeChanges))
        {
            /* Generate the target pathname and write the file */

            OutPathname = calloc (MaxPathLength + strlen (Filename) + 2 + strlen
            if (!OutPathname)
            {
                printf ("Could not allocate buffer for file pathnames\n");
                return (-1);
            }

            strcpy (OutPathname, TargetPath);
            if (SourcePath)
            {
                strcat (OutPathname, "/");
                strcat (OutPathname, Filename);
            }

            AsPutFile (OutPathname, Gbl_FileBuffer, ConversionTable->Flags);
        }
    }

    free (Gbl_FileBuffer);
    free (Pathname);
    if (OutPathname)
    {
        free (OutPathname);
    }

    return (0);
}


/*****************************************************************************
 *
 * FUNCTION:    AsCheckForDirectory
 *
 * DESCRIPTION: Check if the current file is a valid directory. If not,
 *              construct the full pathname for the source and target paths.
 *              Checks for the dot and dot-dot files (they are ignored)
 *
 *****************************************************************************/

ACPI_NATIVE_INT
AsCheckForDirectory (
    char                    *SourceDirPath,
    char                    *TargetDirPath,
    char                    *Filename,
    char                    **SourcePath,
    char                    **TargetPath)
{
    char                    *SrcPath;
    char                    *TgtPath;


    if (!(strcmp (Filename, ".")) ||
```

```
655         !(strcmp (Filename, "..")))
656     {
657         return (-1);
658     }

660     SrcPath = calloc (strlen (SourceDirPath) + strlen (Filename) + 2, 1);
661     if (!SrcPath)
662     {
663         printf ("Could not allocate buffer for directory source pathname\n");
664         return (-1);
665     }

667     TgtPath = calloc (strlen (TargetDirPath) + strlen (Filename) + 2, 1);
668     if (!TgtPath)
669     {
670         printf ("Could not allocate buffer for directory target pathname\n");
671         free (SrcPath);
672         return (-1);
673     }

675     strcpy (SrcPath, SourceDirPath);
676     strcat (SrcPath, "/");
677     strcat (SrcPath, Filename);

679     strcpy (TgtPath, TargetDirPath);
680     strcat (TgtPath, "/");
681     strcat (TgtPath, Filename);

683     *SourcePath = SrcPath;
684     *TargetPath = TgtPath;
685     return (0);
686 }


689 /******************************************************************************
690  *
691  * FUNCTION:    AsGetFile
692  *
693  * DESCRIPTION: Open a file and read it entirely into a an allocated buffer
694  *
695  *****************************************************************************/

697 int
698 AsGetFile (
699     char                    *Filename,
700     char                    **FileBuffer,
701     UINT32                  *FileSize)
702 {
703     FILE                    *File;
704     UINT32                  Size;
705     char                    *Buffer;
706     int                     Seek1;
707     int                     Seek2;
708     size_t                  Actual;


711     /* Binary mode leaves CR/LF pairs */

713     File = fopen (Filename, "rb");
714     if (!File)
715     {
716         printf ("Could not open file %s\n", Filename);
717         return (-1);
718     }

720     /* Need file size to allocate a buffer */
```



```
722     Seek1 = fseek (File, 0L, SEEK_END);
723     Size = ftell (File);
724     Seek2 = fseek (File, 0L, SEEK_SET);

726     if (Seek1 || Seek2 || (Size == -1))
727     {
728         printf ("Could not get file size for %s\n", Filename);
729         goto ErrorExit;
730     }

732     /*
733      * Create a buffer for the entire file
734      * Add plenty extra buffer to accommodate string replacements
735      */
736     Gbl_TotalSize += Size;

738     Buffer = calloc (Size * 2, 1);
739     if (!Buffer)
740     {
741         printf ("Could not allocate buffer of size %u\n", Size * 2);
742         goto ErrorExit;
743     }

745     /* Read the entire file */

747     Actual = fread (Buffer, 1, Size, File);
748     if (Actual != Size)
749     {
750         printf ("Could not read the input file %s (%u bytes)\n",
751             Filename, Size);
752         goto ErrorExit;
753     }

755     Buffer [Size] = 0;          /* Null terminate the buffer */
756     fclose (File);

758     /* Check for unix contamination */

760     Gbl_HasLoneLineFeeds = AsDetectLoneLineFeeds (Filename, Buffer);

762     /*
763      * Convert all CR/LF pairs to LF only. We do this locally so that
764      * this code is portable across operating systems.
765      */
766     AsConvertToLineFeeds (Buffer);

768     *FileBuffer = Buffer;
769     *FileSize = Size;
770     return (0);


773 ErrorExit:

775     fclose (File);
776     return (-1);
777 }


780 /******************************************************************************
781  *
782  * FUNCTION:    AsPutFile
783  *
784  * DESCRIPTION: Create a new output file and write the entire contents of the
785  *              buffer to the new file. Buffer must be a zero terminated string
786  *
```

```
 787  *****************************************************************************/

 789 int
 790 AsPutFile (
 791     char                    *Pathname,
 792     char                    *FileBuffer,
 793     UINT32                  SystemFlags)
 794 {
 795     FILE                    *File;
 796     UINT32                  FileSize;
 797     size_t                  Actual;
 798     int                     Status = 0;


 801     /* Create the target file */

 803     if (!(SystemFlags & FLG_NO_CARRIAGE_RETURNS))
 804     {
 805         /* Put back the CR before each LF */

 807         AsInsertCarriageReturns (FileBuffer);
 808     }

 810     File = fopen (Pathname, "w+b");
 811     if (!File)
 812     {
 813         perror ("Could not create destination file");
 814         printf ("Could not create destination file \"%s\"\n", Pathname);
 815         return (-1);
 816     }

 818     /* Write the buffer to the file */

 820     FileSize = strlen (FileBuffer);
 821     Actual = fwrite (FileBuffer, 1, FileSize, File);
 822     if (Actual != FileSize)
 823     {
 824         printf ("Error writing output file \"%s\"\n", Pathname);
 825         Status = -1;
 826     }

 828     fclose (File);
 829     return (Status);
 830 }
```

```
**********************************************************
   13386 Thu Dec 26 13:50:12 2013
new/usr/src/common/acpica/tools/acpisrc/asmain.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asmain - Main module for the acpi source processor utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpisrc.h"
#include "acapps.h"

/* Local prototypes */

int
AsStricmp (
    char                    *String1,
    char                    *String2);

int
AsExaminePaths (
    ACPI_CONVERSION_TABLE   *ConversionTable,
    char                    *Source,
    char                    *Target,
    UINT32                  *SourceFileType);
```



```
void
AsDisplayStats (
    void);

void
AsDisplayUsage (
    void);

/* Globals */

UINT32                  Gbl_Tabs = 0;
UINT32                  Gbl_MissingBraces = 0;
UINT32                  Gbl_NonAnsiComments = 0;
UINT32                  Gbl_Files = 0;
UINT32                  Gbl_WhiteLines = 0;
UINT32                  Gbl_CommentLines = 0;
UINT32                  Gbl_SourceLines = 0;
UINT32                  Gbl_LongLines = 0;
UINT32                  Gbl_TotalLines = 0;
UINT32                  Gbl_TotalSize = 0;
UINT32                  Gbl_HeaderLines = 0;
UINT32                  Gbl_HeaderSize = 0;
void                    *Gbl_StructDefs = NULL;

struct stat             Gbl_StatBuf;
char                    *Gbl_FileBuffer;
UINT32                  Gbl_FileSize;
UINT32                  Gbl_FileType;
BOOLEAN                 Gbl_VerboseMode = FALSE;
BOOLEAN                 Gbl_QuietMode = FALSE;
BOOLEAN                 Gbl_BatchMode = FALSE;
BOOLEAN                 Gbl_DebugStatementsMode = FALSE;
BOOLEAN                 Gbl_MadeChanges = FALSE;
BOOLEAN                 Gbl_Overwrite = FALSE;
BOOLEAN                 Gbl_WidenDeclarations = FALSE;
BOOLEAN                 Gbl_IgnoreLoneLineFeeds = FALSE;
BOOLEAN                 Gbl_HasLoneLineFeeds = FALSE;
BOOLEAN                 Gbl_Cleanup = FALSE;
BOOLEAN                 Gbl_IgnoreTranslationEscapes = FALSE;

#define AS_UTILITY_NAME             "ACPI Source Code Conversion Utility"
#define AS_SUPPORTED_OPTIONS        "cdhilqsuv^y"


/******************************************************************************
 *
 * FUNCTION:    AsStricmp
 *
 * DESCRIPTION: Implementation of the non-ANSI stricmp function (compare
 *              strings with no case sensitivity)
 *
 *****************************************************************************/

int
AsStricmp (
    char                    *String1,
    char                    *String2)
{
    int                     c1;
    int                     c2;


    do
    {
        c1 = tolower ((int) *String1);
```

```
127         c2 = tolower ((int) *String2);

129         String1++;
130         String2++;
131     }
132     while ((c1 == c2) && (c1));

134     return (c1 - c2);
135 }


138 /******************************************************************************
139  *
140  * FUNCTION:    AsExaminePaths
141  *
142  * DESCRIPTION: Source and Target pathname verification and handling
143  *
144  *****************************************************************************/

146 int
147 AsExaminePaths (
148     ACPI_CONVERSION_TABLE   *ConversionTable,
149     char                    *Source,
150     char                    *Target,
151     UINT32                  *SourceFileType)
152 {
153     int                     Status;
154     int                     Response;


157     Status = stat (Source, &Gbl_StatBuf);
158     if (Status)
159     {
160         printf ("Source path \"%s\" does not exist\n", Source);
161         return (-1);
162     }

164     /* Return the filetype -- file or a directory */

166     *SourceFileType = 0;
167     if (Gbl_StatBuf.st_mode & S_IFDIR)
168     {
169         *SourceFileType = S_IFDIR;
170     }

172     /*
173      * If we are in no-output mode or in batch mode, we are done
174      */
175     if ((ConversionTable->Flags & FLG_NO_FILE_OUTPUT) ||
176         (Gbl_BatchMode))
177     {
178         return (0);
179     }

181     if (!AsStricmp (Source, Target))
182     {
183         printf ("Target path is the same as the source path, overwrite?\n");
184         Response = getchar ();

186         /* Check response */

188         if (Response != 'y')
189         {
190             return (-1);
191         }
```



```
193         Gbl_Overwrite = TRUE;
194     }
195     else
196     {
197         Status = stat (Target, &Gbl_StatBuf);
198         if (!Status)
199         {
200             printf ("Target path already exists, overwrite?\n");
201             Response = getchar ();

203             /* Check response */

205             if (Response != 'y')
206             {
207                 return (-1);
208             }
209         }
210     }

212     return (0);
213 }


216 /******************************************************************************
217  *
218  * FUNCTION:    AsDisplayStats
219  *
220  * DESCRIPTION: Display global statistics gathered during translation
221  *
222  *****************************************************************************/

224 void
225 AsDisplayStats (
226     void)
227 {

229     if (Gbl_QuietMode)
230     {
231         return;
232     }

234     printf ("\nAcpiSrc statistics:\n\n");
235     printf ("%8u Files processed\n", Gbl_Files);

237     if (!Gbl_Files)
238     {
239         return;
240     }

242     printf ("%8u Total bytes (%.1fK/file)\n",
243         Gbl_TotalSize, ((double) Gbl_TotalSize/Gbl_Files)/1024);
244     printf ("%8u Tabs found\n", Gbl_Tabs);
245     printf ("%8u Missing if/else braces\n", Gbl_MissingBraces);
246     printf ("%8u Non-ANSI comments found\n", Gbl_NonAnsiComments);
247     printf ("%8u Total Lines\n", Gbl_TotalLines);
248     printf ("%8u Lines of code\n", Gbl_SourceLines);
249     printf ("%8u Lines of non-comment whitespace\n", Gbl_WhiteLines);
250     printf ("%8u Lines of comments\n", Gbl_CommentLines);
251     printf ("%8u Long lines found\n", Gbl_LongLines);

253     if (Gbl_WhiteLines > 0)
254     {
255         printf ("%8.1f Ratio of code to whitespace\n",
256             ((float) Gbl_SourceLines / (float) Gbl_WhiteLines));
257     }
```

```
259     if ((Gbl_CommentLines + Gbl_NonAnsiComments) > 0)
260     {
261         printf ("%8.1f Ratio of code to comments\n",
262             ((float) Gbl_SourceLines / (float) (Gbl_CommentLines + Gbl_NonAnsiCo
263     }

265     if (!Gbl_TotalLines)
266     {
267         return;
268     }

270     printf ("         %u%% code, %u%% comments, %u%% whitespace, %u%% headers\n"
271         (Gbl_SourceLines * 100) / Gbl_TotalLines,
272         (Gbl_CommentLines * 100) / Gbl_TotalLines,
273         (Gbl_WhiteLines * 100) / Gbl_TotalLines,
274         (Gbl_HeaderLines * 100) / Gbl_TotalLines);
275     return;
276 }


279 /******************************************************************************
280  *
281  * FUNCTION:    AsDisplayUsage
282  *
283  * DESCRIPTION: Usage message
284  *
285  *****************************************************************************/

287 void
288 AsDisplayUsage (
289     void)
290 {

292     ACPI_USAGE_HEADER ("acpisrc [-c|l|u] [-dsvy] <SourceDir> <DestinationDir>");

294     ACPI_OPTION ("-c",          "Generate cleaned version of the source");
295     ACPI_OPTION ("-h",          "Insert dual-license header into all modules");
296     ACPI_OPTION ("-i",          "Cleanup macro indentation");
297     ACPI_OPTION ("-l",          "Generate Linux version of the source");
298     ACPI_OPTION ("-u",          "Generate Custom source translation");

300     printf ("\n");
301     ACPI_OPTION ("-d",          "Leave debug statements in code");
302     ACPI_OPTION ("-s",          "Generate source statistics only");
303     ACPI_OPTION ("-v",          "Display version information");
304     ACPI_OPTION ("-vb",         "Verbose mode");
305     ACPI_OPTION ("-y",          "Suppress file overwrite prompts");
306 }


309 /******************************************************************************
310  *
311  * FUNCTION:    main
312  *
313  * DESCRIPTION: C main function
314  *
315  *****************************************************************************/

317 int ACPI_SYSTEM_XFACE
318 main (
319     int                     argc,
320     char                    *argv[])
321 {
322     int                     j;
323     ACPI_CONVERSION_TABLE   *ConversionTable = NULL;
324     char                    *SourcePath;
```



```
325     char                    *TargetPath;
326     UINT32                  FileType;


329     ACPI_DEBUG_INITIALIZE (); /* For debug version only */
330     printf (ACPI_COMMON_SIGNON (AS_UTILITY_NAME));

332     if (argc < 2)
333     {
334         AsDisplayUsage ();
335         return (0);
336     }

338     /* Command line options */

340     while ((j = AcpiGetopt (argc, argv, AS_SUPPORTED_OPTIONS)) != EOF) switch(j)
341     {
342     case 'l':

344         /* Linux code generation */

346         printf ("Creating Linux source code\n");
347         ConversionTable = &LinuxConversionTable;
348         Gbl_WidenDeclarations = TRUE;
349         Gbl_IgnoreLoneLineFeeds = TRUE;
350         break;

352     case 'c':

354         /* Cleanup code */

356         printf ("Code cleanup\n");
357         ConversionTable = &CleanupConversionTable;
358         Gbl_Cleanup = TRUE;
359         break;

361     case 'h':

363         /* Inject Dual-license header */

365         printf ("Inserting Dual-license header to all modules\n");
366         ConversionTable = &LicenseConversionTable;
367         break;

369     case 'i':

371         /* Cleanup wrong indent result */

373         printf ("Cleaning up macro indentation\n");
374         ConversionTable = &IndentConversionTable;
375         Gbl_IgnoreLoneLineFeeds = TRUE;
376         Gbl_IgnoreTranslationEscapes = TRUE;
377         break;

379     case 's':

381         /* Statistics only */

383         break;

385     case 'u':

387         /* custom conversion  */

389         printf ("Custom source translation\n");
390         ConversionTable = &CustomConversionTable;
```

```
391         break;

393     case 'v':

395         switch (AcpiGbl_Optarg[0])
396         {
397         case '^':  /* -v: (Version): signon already emitted, just exit */

399             exit (0);

401         case 'b':

403             /* Verbose mode */

405             Gbl_VerboseMode = TRUE;
406             break;

408         default:

410             printf ("Unknown option: -v%s\n", AcpiGbl_Optarg);
411             return (-1);
412         }

414         break;

416     case 'y':

418         /* Batch mode */

420         Gbl_BatchMode = TRUE;
421         break;

423     case 'd':

425         /* Leave debug statements in */

427         Gbl_DebugStatementsMode = TRUE;
428         break;

430     case 'q':

432         /* Quiet mode */

434         Gbl_QuietMode = TRUE;
435         break;

437     default:

439         AsDisplayUsage ();
440         return (-1);
441     }


444     SourcePath = argv[AcpiGbl_Optind];
445     if (!SourcePath)
446     {
447         printf ("Missing source path\n");
448         AsDisplayUsage ();
449         return (-1);
450     }

452     TargetPath = argv[AcpiGbl_Optind+1];

454     if (!ConversionTable)
455     {
456         /* Just generate statistics. Ignore target path */
```



```
458         TargetPath = SourcePath;

460         printf ("Source code statistics only\n");
461         ConversionTable = &StatsConversionTable;
462     }
463     else if (!TargetPath)
464     {
465         TargetPath = SourcePath;
466     }

468     if (Gbl_DebugStatementsMode)
469     {
470         ConversionTable->SourceFunctions &= ~CVT_REMOVE_DEBUG_MACROS;
471     }

473     /* Check source and target paths and files */

475     if (AsExaminePaths (ConversionTable, SourcePath, TargetPath, &FileType))
476     {
477         return (-1);
478     }

480     /* Source/target can be either directories or a files */

482     if (FileType == S_IFDIR)
483     {
484         /* Process the directory tree */

486         AsProcessTree (ConversionTable, SourcePath, TargetPath);
487     }
488     else
489     {
490         /* Process a single file */

492         /* Differentiate between source and header files */

494         if (strstr (SourcePath, ".h"))
495         {
496             AsProcessOneFile (ConversionTable, NULL, TargetPath, 0, SourcePath,
497         }
498         else
499         {
500             AsProcessOneFile (ConversionTable, NULL, TargetPath, 0, SourcePath,
501         }
502     }

504     /* Always display final summary and stats */

506     AsDisplayStats ();

508     return (0);
509 }
```

```
**********************************************************
   19063 Thu Dec 26 13:50:12 2013
new/usr/src/common/acpica/tools/acpisrc/asremove.c
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
/******************************************************************************
 *
 * Module Name: asremove - Source conversion - removal functions
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpisrc.h"

/* Local prototypes */

void
AsRemoveStatement (
    char                    *Buffer,
    char                    *Keyword,
    UINT32                  Type);


/******************************************************************************
 *
 * FUNCTION:    AsRemoveStatement
 *
 * DESCRIPTION: Remove all statements that contain the given keyword.
```



```
 *              Limitations:  Removes text from the start of the line that
 *              contains the keyword to the next semicolon. Currently
 *              doesn't ignore comments.
 *
 *****************************************************************************/

void
AsRemoveStatement (
    char                    *Buffer,
    char                    *Keyword,
    UINT32                  Type)
{
    char                    *SubString;
    char                    *SubBuffer;
    int                     KeywordLength;


    KeywordLength = strlen (Keyword);
    SubBuffer = Buffer;
    SubString = Buffer;


    while (SubString)
    {
        SubString = strstr (SubBuffer, Keyword);

        if (SubString)
        {
            SubBuffer = SubString;

            if ((Type == REPLACE_WHOLE_WORD) &&
                (!AsMatchExactWord (SubString, KeywordLength)))
            {
                SubBuffer++;
                continue;
            }

            /* Find start of this line */

            while (*SubString != '\n')
            {
                SubString--;
            }
            SubString++;

            /* Find end of this statement */

            SubBuffer = AsSkipPastChar (SubBuffer, ';');
            if (!SubBuffer)
            {
                return;
            }

            /* Find end of this line */

            SubBuffer = AsSkipPastChar (SubBuffer, '\n');
            if (!SubBuffer)
            {
                return;
            }

            /* If next line is blank, remove it too */

            if (*SubBuffer == '\n')
            {
                SubBuffer++;
```

```
126             }
128             /* Remove the lines */
130             SubBuffer = AsRemoveData (SubString, SubBuffer);
131         }
132     }
133 }


136 /******************************************************************************
137  *
138  * FUNCTION:    AsRemoveConditionalCompile
139  *
140  * DESCRIPTION: Remove a "#ifdef" statement, and all text that it encompasses.
141  *              Limitations: cannot handle nested ifdefs.
142  *
143  *****************************************************************************/

145 void
146 AsRemoveConditionalCompile (
147     char                    *Buffer,
148     char                    *Keyword)
149 {
150     char                    *SubString;
151     char                    *SubBuffer;
152     char                    *IfPtr;
153     char                    *EndifPtr;
154     char                    *ElsePtr;
155     char                    *Comment;
156     int                     KeywordLength;


159     KeywordLength = strlen (Keyword);
160     SubBuffer = Buffer;
161     SubString = Buffer;


164     while (SubString)
165     {
166         SubBuffer = strstr (SubString, Keyword);
167         if (!SubBuffer)
168         {
169             return;
170         }

172         /*
173          * Check for translation escape string -- means to ignore
174          * blocks of code while replacing
175          */
176         if (Gbl_IgnoreTranslationEscapes)
177         {
178             Comment = NULL;
179         }
180         else
181         {
182             Comment = strstr (SubString, AS_START_IGNORE);
183         }

185         if ((Comment) &&
186             (Comment < SubBuffer))
187         {
188             SubString = strstr (Comment, AS_STOP_IGNORE);
189             if (!SubString)
190             {
191                 return;
```



```
192             }

194             SubString += 3;
195             continue;
196         }

198         /* Check for ordinary comment */

200         Comment = strstr (SubString, "/*");

202         if ((Comment) &&
203             (Comment < SubBuffer))
204         {
205             SubString = strstr (Comment, "*/");
206             if (!SubString)
207             {
208                 return;
209             }

211             SubString += 2;
212             continue;
213         }

215         SubString = SubBuffer;
216         if (!AsMatchExactWord (SubString, KeywordLength))
217         {
218             SubString++;
219             continue;
220         }

222         /* Find start of this line */

224         while (*SubString != '\n' && (SubString > Buffer))
225         {
226             SubString--;
227         }
228         SubString++;

230         /* Find the "#ifxxxx" */

232         IfPtr = strstr (SubString, "#if");
233         if (!IfPtr)
234         {
235             return;
236         }

238         if (IfPtr > SubBuffer)
239         {
240             /* Not the right #if */

242             SubString = SubBuffer + strlen (Keyword);
243             continue;
244         }

246         /* Find closing #endif or #else */

248         EndifPtr = strstr (SubBuffer, "#endif");
249         if (!EndifPtr)
250         {
251             /* There has to be an #endif */

253             return;
254         }

256         ElsePtr = strstr (SubBuffer, "#else");
257         if ((ElsePtr) &&
```

```
258             (EndifPtr > ElsePtr))
259         {
260             /* This #ifdef contains an #else clause */
261             /* Find end of this line */

263             SubBuffer = AsSkipPastChar (ElsePtr, '\n');
264             if (!SubBuffer)
265             {
266                 return;
267             }

269             /* Remove the #ifdef .... #else code */

271             AsRemoveData (SubString, SubBuffer);

273             /* Next, we will remove the #endif statement */

275             EndifPtr = strstr (SubString, "#endif");
276             if (!EndifPtr)
277             {
278                 /* There has to be an #endif */

280                 return;
281             }

283             SubString = EndifPtr;
284         }

286         /* Remove the ... #endif part */
287         /* Find end of this line */

289         SubBuffer = AsSkipPastChar (EndifPtr, '\n');
290         if (!SubBuffer)
291         {
292             return;
293         }

295         /* Remove the lines */

297         SubBuffer = AsRemoveData (SubString, SubBuffer);
298     }
299 }


302 /******************************************************************************
303  *
304  * FUNCTION:    AsRemoveMacro
305  *
306  * DESCRIPTION: Remove every line that contains the keyword. Does not
307  *              skip comments.
308  *
309  *****************************************************************************/

311 void
312 AsRemoveMacro (
313     char                    *Buffer,
314     char                    *Keyword)
315 {
316     char                    *SubString;
317     char                    *SubBuffer;
318     int                     NestLevel;


321     SubBuffer = Buffer;
322     SubString = Buffer;
```



```
325     while (SubString)
326     {
327         SubString = strstr (SubBuffer, Keyword);

329         if (SubString)
330         {
331             SubBuffer = SubString;

333             /* Find start of the macro parameters */

335             while (*SubString != '(')
336             {
337                 SubString++;
338             }
339             SubString++;

341             /* Remove the macro name and opening paren */

343             SubString = AsRemoveData (SubBuffer, SubString);

345             NestLevel = 1;
346             while (*SubString)
347             {
348                 if (*SubString == '(')
349                 {
350                     NestLevel++;
351                 }
352                 else if (*SubString == ')')
353                 {
354                     NestLevel--;
355                 }

357                 SubString++;

359                 if (NestLevel == 0)
360                 {
361                     break;
362                 }
363             }

365             /* Remove the closing paren */

367             SubBuffer = AsRemoveData (SubString-1, SubString);
368         }
369     }
370 }


373 /******************************************************************************
374  *
375  * FUNCTION:    AsRemoveLine
376  *
377  * DESCRIPTION: Remove every line that contains the keyword. Does not
378  *              skip comments.
379  *
380  *****************************************************************************/

382 void
383 AsRemoveLine (
384     char                    *Buffer,
385     char                    *Keyword)
386 {
387     char                    *SubString;
388     char                    *SubBuffer;
```

```
391     SubBuffer = Buffer;
392     SubString = Buffer;


395     while (SubString)
396     {
397         SubString = strstr (SubBuffer, Keyword);

399         if (SubString)
400         {
401             SubBuffer = SubString;

403             /* Find start of this line */

405             while (*SubString != '\n')
406             {
407                 SubString--;
408             }
409             SubString++;

411             /* Find end of this line */

413             SubBuffer = AsSkipPastChar (SubBuffer, '\n');
414             if (!SubBuffer)
415             {
416                 return;
417             }

419             /* Remove the line */

421             SubBuffer = AsRemoveData (SubString, SubBuffer);
422         }
423     }
424 }


427 /******************************************************************************
428  *
429  * FUNCTION:    AsReduceTypedefs
430  *
431  * DESCRIPTION: Eliminate certain typedefs
432  *
433  *****************************************************************************/

435 void
436 AsReduceTypedefs (
437     char                    *Buffer,
438     char                    *Keyword)
439 {
440     char                    *SubString;
441     char                    *SubBuffer;
442     int                     NestLevel;


445     SubBuffer = Buffer;
446     SubString = Buffer;


449     while (SubString)
450     {
451         SubString = strstr (SubBuffer, Keyword);

453         if (SubString)
454         {
455             /* Remove the typedef itself */
```



```
457             SubBuffer = SubString + strlen ("typedef") + 1;
458             SubBuffer = AsRemoveData (SubString, SubBuffer);

460             /* Find the opening brace of the struct or union */

462             while (*SubString != '{')
463             {
464                 SubString++;
465             }
466             SubString++;

468             /* Find the closing brace. Handles nested braces */

470             NestLevel = 1;
471             while (*SubString)
472             {
473                 if (*SubString == '{')
474                 {
475                     NestLevel++;
476                 }
477                 else if (*SubString == '}')
478                 {
479                     NestLevel--;
480                 }

482                 SubString++;

484                 if (NestLevel == 0)
485                 {
486                     break;
487                 }
488             }

490             /* Remove an extra line feed if present */

492             if (!strncmp (SubString - 3, "\n\n", 2))
493             {
494                 *(SubString -2) = '}';
495                 SubString--;
496             }

498             /* Find the end of the typedef name */

500             SubBuffer = AsSkipUntilChar (SubString, ';');

502             /* And remove the typedef name */

504             SubBuffer = AsRemoveData (SubString, SubBuffer);
505         }
506     }
507 }


510 /******************************************************************************
511  *
512  * FUNCTION:    AsRemoveEmptyBlocks
513  *
514  * DESCRIPTION: Remove any C blocks (e.g., if {}) that contain no code. This
515  *              can happen as a result of removing lines such as DEBUG_PRINT.
516  *
517  *****************************************************************************/

519 void
520 AsRemoveEmptyBlocks (
521     char                    *Buffer,
```

```
522     char                    *Filename)
523 {
524     char                    *SubBuffer;
525     char                    *BlockStart;
526     BOOLEAN                 EmptyBlock = TRUE;
527     BOOLEAN                 AnotherPassRequired = TRUE;
528     UINT32                  BlockCount = 0;


531     while (AnotherPassRequired)
532     {
533         SubBuffer = Buffer;
534         AnotherPassRequired = FALSE;

536         while (*SubBuffer)
537         {
538             if (*SubBuffer == '{')
539             {
540                 BlockStart = SubBuffer;
541                 EmptyBlock = TRUE;

543                 SubBuffer++;
544                 while (*SubBuffer != '}')
545                 {
546                     if ((*SubBuffer != ' ') &&
547                         (*SubBuffer != '\n'))
548                     {
549                         EmptyBlock = FALSE;
550                         break;
551                     }
552                     SubBuffer++;
553                 }

555                 if (EmptyBlock)
556                 {
557                     /* Find start of the first line of the block */

559                     while (*BlockStart != '\n')
560                     {
561                         BlockStart--;
562                     }

564                     /* Find end of the last line of the block */

566                     SubBuffer = AsSkipUntilChar (SubBuffer, '\n');
567                     if (!SubBuffer)
568                     {
569                         break;
570                     }

572                     /* Remove the block */

574                     SubBuffer = AsRemoveData (BlockStart, SubBuffer);
575                     BlockCount++;
576                     AnotherPassRequired = TRUE;
577                     continue;
578                 }
579             }

581             SubBuffer++;
582         }
583     }

585     if (BlockCount)
586     {
587         Gbl_MadeChanges = TRUE;
```



```
588         AsPrint ("Code blocks deleted", BlockCount, Filename);
589     }
590 }


593 /******************************************************************************
594  *
595  * FUNCTION:    AsRemoveDebugMacros
596  *
597  * DESCRIPTION: Remove all "Debug" macros -- macros that produce debug output.
598  *
599  *****************************************************************************/

601 void
602 AsRemoveDebugMacros (
603     char                    *Buffer)
604 {
605     AsRemoveConditionalCompile (Buffer, "ACPI_DEBUG_OUTPUT");

607     AsRemoveStatement (Buffer, "ACPI_DEBUG_PRINT",      REPLACE_WHOLE_WORD);
608     AsRemoveStatement (Buffer, "ACPI_DEBUG_PRINT_RAW",  REPLACE_WHOLE_WORD);
609     AsRemoveStatement (Buffer, "DEBUG_EXEC",            REPLACE_WHOLE_WORD);
610     AsRemoveStatement (Buffer, "FUNCTION_ENTRY",        REPLACE_WHOLE_WORD);
611     AsRemoveStatement (Buffer, "PROC_NAME",             REPLACE_WHOLE_WORD);
612     AsRemoveStatement (Buffer, "FUNCTION_TRACE",        REPLACE_SUBSTRINGS);
613     AsRemoveStatement (Buffer, "DUMP_",                 REPLACE_SUBSTRINGS);

615     AsReplaceString ("return_VOID",         "return", REPLACE_WHOLE_WORD, Buffer
616     AsReplaceString ("return_PTR",          "return", REPLACE_WHOLE_WORD, Buffer
617     AsReplaceString ("return_ACPI_STATUS",  "return", REPLACE_WHOLE_WORD, Buffer
618     AsReplaceString ("return_acpi_status",  "return", REPLACE_WHOLE_WORD, Buffer
619     AsReplaceString ("return_VALUE",        "return", REPLACE_WHOLE_WORD, Buffer
620 }


623 /******************************************************************************
624  *
625  * FUNCTION:    AsCleanupSpecialMacro
626  *
627  * DESCRIPTION: For special macro invocations (invoked without ";" at the end
628  *              of the lines), do the following:
629  *              1. Remove spaces appended by indent at the beginning of lines.
630  *              2. Add an empty line between two special macro invocations.
631  *
632  *****************************************************************************/

634 void
635 AsCleanupSpecialMacro (
636     char                    *Buffer,
637     char                    *Keyword)
638 {
639     char                    *SubString;
640     char                    *SubBuffer;
641     char                    *CommentEnd;
642     int                     NewLine;
643     int                     NestLevel;


646     SubBuffer = Buffer;
647     SubString = Buffer;

649     while (SubString)
650     {
651         SubString = strstr (SubBuffer, Keyword);

653         if (SubString)
```

```
654         {
655             /* Find start of the macro parameters */

657             while (*SubString != '(')
658             {
659                 SubString++;
660             }
661             SubString++;

663             NestLevel = 1;
664             while (*SubString)
665             {
666                 if (*SubString == '(')
667                 {
668                     NestLevel++;
669                 }
670                 else if (*SubString == ')')
671                 {
672                     NestLevel--;
673                 }

675                 SubString++;

677                 if (NestLevel == 0)
678                 {
679                     break;
680                 }
681             }

683 SkipLine:

685             /* Find end of the line */

687             NewLine = FALSE;
688             while (!NewLine && *SubString)
689             {
690                 if (*SubString == '\n' && *(SubString - 1) != '\\')
691                 {
692                     NewLine = TRUE;
693                 }
694                 SubString++;
695             }

697             /* Find end of the line */

699             if (*SubString == '#' || *SubString == '\n')
700             {
701                 goto SkipLine;
702             }

704             SubBuffer = SubString;

706             /* Find start of the non-space */

708             while (*SubString == ' ')
709             {
710                 SubString++;
711             }

713             /* Find end of the line */

715             if (*SubString == '#' || *SubString == '\n')
716             {
717                 goto SkipLine;
718             }
```



```
720             /* Find end of the line */

722             if (*SubString == '/' || *SubString == '*')
723             {
724                 CommentEnd = strstr (SubString, "*/");
725                 if (CommentEnd)
726                 {
727                     SubString = CommentEnd + 2;
728                     goto SkipLine;
729                 }
730             }

732             SubString = AsRemoveData (SubBuffer, SubString);
733         }
734     }
735 }
```

```
**********************************************************
   46557 Thu Dec 26 13:50:13 2013
new/usr/src/common/acpica/tools/acpisrc/astable.c
update to acpica-unix2-20130927
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: astable - Tables used for source conversion
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */


#include "acpisrc.h"
#include "acapps.h"


/******************************************************************************
 *
 * Standard/Common translation tables
 *
 *****************************************************************************/


ACPI_STRING_TABLE           StandardDataTypes[] = {

    /* Declarations first */

    {"UINT32      ",     "unsigned int",     REPLACE_SUBSTRINGS},
```



```
    {"UINT16      ",     "unsigned short",   REPLACE_SUBSTRINGS},
    {"UINT8       ",     "unsigned char",    REPLACE_SUBSTRINGS},
    {"BOOLEAN     ",     "unsigned char",    REPLACE_SUBSTRINGS},

    /* Now do embedded typecasts */

    {"UINT32",           "unsigned int",     REPLACE_SUBSTRINGS},
    {"UINT16",           "unsigned short",   REPLACE_SUBSTRINGS},
    {"UINT8",            "unsigned char",    REPLACE_SUBSTRINGS},
    {"BOOLEAN",          "unsigned char",    REPLACE_SUBSTRINGS},

    {"INT32  ",          "int    ",          REPLACE_SUBSTRINGS},
    {"INT32",            "int",              REPLACE_SUBSTRINGS},
    {"INT16",            "short",            REPLACE_SUBSTRINGS},
    {"INT8",             "char",             REPLACE_SUBSTRINGS},

    /* Put back anything we broke (such as anything with _INT32_ in it) */

    {"_int_",            "_INT32_",          REPLACE_SUBSTRINGS},
    {"_unsigned int_",   "_UINT32_",         REPLACE_SUBSTRINGS},
    {NULL,               NULL,               0}
};


/******************************************************************************
 *
 * Linux-specific translation tables
 *
 *****************************************************************************/

char                        DualLicenseHeader[] =
"/*\n"
" * Copyright (C) 2000 - 2013, Intel Corp.\n"
" * All rights reserved.\n"
" *\n"
" * Redistribution and use in source and binary forms, with or without\n"
" * modification, are permitted provided that the following conditions\n"
" * are met:\n"
" * 1. Redistributions of source code must retain the above copyright\n"
" *    notice, this list of conditions, and the following disclaimer,\n"
" *    without modification.\n"
" * 2. Redistributions in binary form must reproduce at minimum a disclaimer\n"
" *    substantially similar to the \"NO WARRANTY\" disclaimer below\n"
" *    (\"Disclaimer\") and any redistribution must be conditioned upon\n"
" *    including a substantially similar Disclaimer requirement for further\n"
" *    binary redistribution.\n"
" * 3. Neither the names of the above-listed copyright holders nor the names\n"
" *    of any contributors may be used to endorse or promote products derived\n"
" *    from this software without specific prior written permission.\n"
" *\n"
" * Alternatively, this software may be distributed under the terms of the\n"
" * GNU General Public License (\"GPL\") version 2 as published by the Free\n"
" * Software Foundation.\n"
" *\n"
" * NO WARRANTY\n"
" * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS\n"
" * \"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT\n"
" * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR\n"
" * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT\n"
" * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL\n"
" * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS\n"
" * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n"
" * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,\n"
" * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING\n"
" * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE\n"
" * POSSIBILITY OF SUCH DAMAGES.\n"
```

```
" */\n";

ACPI_STRING_TABLE           LinuxDataTypes[] = {

/*
 * Extra space is added after the type so there is room to add "struct", "union"
 * etc. when the existing struct typedefs are eliminated.
 */

    /* Declarations first - ACPI types and standard C types */

    {"INT64       ",            "s64         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"UINT64      ",            "u64         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"UINT32      ",            "u32         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"INT32       ",            "s32         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"UINT16      ",            "u16         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"INT16       ",            "s16         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"UINT8       ",            "u8          ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"BOOLEAN     ",            "u8          ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"char        ",            "char        ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"void        ",            "void        ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"char *      ",            "char *      ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"void *      ",            "void *      ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"int         ",            "int         ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"FILE        ",            "FILE        ",     REPLACE_WHOLE_WORD | EXTRA_I
    {"size_t      ",            "size_t      ",     REPLACE_WHOLE_WORD | EXTRA_I

    /* Now do embedded typecasts */

    {"UINT64",                  "u64",              REPLACE_WHOLE_WORD},
    {"UINT32",                  "u32",              REPLACE_WHOLE_WORD},
    {"UINT16",                  "u16",              REPLACE_WHOLE_WORD},
    {"UINT8",                   "u8",               REPLACE_WHOLE_WORD},
    {"BOOLEAN",                 "u8",               REPLACE_WHOLE_WORD},

    {"INT64  ",                 "s64    ",          REPLACE_WHOLE_WORD},
    {"INT64",                   "s64",              REPLACE_WHOLE_WORD},
    {"INT32  ",                 "s32    ",          REPLACE_WHOLE_WORD},
    {"INT32",                   "s32",              REPLACE_WHOLE_WORD},
    {"INT16  ",                 "s16    ",          REPLACE_WHOLE_WORD},
    {"INT8   ",                 "s8     ",          REPLACE_WHOLE_WORD},
    {"INT16",                   "s16",              REPLACE_WHOLE_WORD},
    {"INT8",                    "s8",               REPLACE_WHOLE_WORD},

    {NULL,                      NULL,               0}
};

ACPI_TYPED_IDENTIFIER_TABLE           AcpiIdentifiers[] = {

    {"ACPI_ADDRESS_RANGE",              SRC_TYPE_STRUCT},
    {"ACPI_ADR_SPACE_HANDLER",          SRC_TYPE_SIMPLE},
    {"ACPI_ADR_SPACE_SETUP",            SRC_TYPE_SIMPLE},
    {"ACPI_ADR_SPACE_TYPE",             SRC_TYPE_SIMPLE},
    {"ACPI_AML_OPERANDS",               SRC_TYPE_UNION},
    {"ACPI_BIT_REGISTER_INFO",          SRC_TYPE_STRUCT},
    {"ACPI_BUFFER",                     SRC_TYPE_STRUCT},
    {"ACPI_BUS_ATTRIBUTE",              SRC_TYPE_STRUCT},
    {"ACPI_CACHE_T",                    SRC_TYPE_SIMPLE},
    {"ACPI_CMTABLE_HANDLER",            SRC_TYPE_SIMPLE},
    {"ACPI_COMMON_FACS",                SRC_TYPE_STRUCT},
    {"ACPI_COMMON_STATE",               SRC_TYPE_STRUCT},
    {"ACPI_COMMON_DESCRIPTOR",          SRC_TYPE_STRUCT},
    {"ACPI_COMPATIBLE_ID",              SRC_TYPE_STRUCT},
    {"ACPI_CONNECTION_INFO",            SRC_TYPE_STRUCT},
    {"ACPI_CONTROL_STATE",              SRC_TYPE_STRUCT},
    {"ACPI_CONVERSION_TABLE",           SRC_TYPE_STRUCT},
```



```
    {"ACPI_CPU_FLAGS",                  SRC_TYPE_SIMPLE},
    {"ACPI_CREATE_FIELD_INFO",          SRC_TYPE_STRUCT},
    {"ACPI_DB_ARGUMENT_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_DB_COMMAND_HELP",            SRC_TYPE_STRUCT},
    {"ACPI_DB_COMMAND_INFO",            SRC_TYPE_STRUCT},
    {"ACPI_DB_EXECUTE_WALK",            SRC_TYPE_STRUCT},
    {"ACPI_DB_METHOD_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_DEBUG_MEM_BLOCK",            SRC_TYPE_STRUCT},
    {"ACPI_DEBUG_MEM_HEADER",           SRC_TYPE_STRUCT},
    {"ACPI_DEBUG_PRINT_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_DESCRIPTOR",                 SRC_TYPE_UNION},
    {"ACPI_DEVICE_INFO",                SRC_TYPE_STRUCT},
    {"ACPI_DEVICE_WALK_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_DMTABLE_DATA",               SRC_TYPE_STRUCT},
    {"ACPI_DMTABLE_INFO",               SRC_TYPE_STRUCT},
    {"ACPI_DMTABLE_HANDLER",            SRC_TYPE_SIMPLE},
    {"ACPI_EVALUATE_INFO",              SRC_TYPE_STRUCT},
    {"ACPI_EVENT_HANDLER",              SRC_TYPE_SIMPLE},
    {"ACPI_EVENT_STATUS",               SRC_TYPE_SIMPLE},
    {"ACPI_EVENT_TYPE",                 SRC_TYPE_SIMPLE},
    {"ACPI_EXCEPTION_HANDLER",          SRC_TYPE_SIMPLE},
    {"ACPI_EXCEPTION_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_EXDUMP_INFO",                SRC_TYPE_STRUCT},
    {"ACPI_EXECUTE_OP",                 SRC_TYPE_SIMPLE},
    {"ACPI_EXECUTE_TYPE",               SRC_TYPE_SIMPLE},
    {"ACPI_EXTERNAL_LIST",              SRC_TYPE_STRUCT},
    {"ACPI_EXTERNAL_FILE",              SRC_TYPE_STRUCT},
    {"ACPI_FADT_INFO",                  SRC_TYPE_STRUCT},
    {"ACPI_FADT_PM_INFO",               SRC_TYPE_STRUCT},
    {"ACPI_FIELD_INFO",                 SRC_TYPE_STRUCT},
    {"ACPI_FIND_CONTEXT",               SRC_TYPE_STRUCT},
    {"ACPI_FIXED_EVENT_HANDLER",        SRC_TYPE_STRUCT},
    {"ACPI_FIXED_EVENT_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_GBL_EVENT_HANDLER",          SRC_TYPE_SIMPLE},
    {"ACPI_GENERIC_ADDRESS",            SRC_TYPE_STRUCT},
    {"ACPI_GENERIC_STATE",              SRC_TYPE_UNION},
    {"ACPI_GET_DEVICES_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_GLOBAL_NOTIFY_HANDLER",      SRC_TYPE_STRUCT},
    {"ACPI_GPE_BLOCK_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_GPE_CALLBACK",               SRC_TYPE_SIMPLE},
    {"ACPI_GPE_DEVICE_INFO",            SRC_TYPE_STRUCT},
    {"ACPI_GPE_EVENT_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_GPE_HANDLER",                SRC_TYPE_SIMPLE},
    {"ACPI_GPE_HANDLER_INFO",           SRC_TYPE_STRUCT},
    {"ACPI_GPE_INDEX_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_GPE_NOTIFY_INFO",            SRC_TYPE_STRUCT},
    {"ACPI_GPE_REGISTER_INFO",          SRC_TYPE_STRUCT},
    {"ACPI_GPE_WALK_INFO",              SRC_TYPE_STRUCT},
    {"ACPI_GPE_XRUPT_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_HANDLE",                     SRC_TYPE_SIMPLE},
    {"ACPI_HANDLER_INFO",               SRC_TYPE_STRUCT},
    {"ACPI_INIT_HANDLER",               SRC_TYPE_SIMPLE},
    {"ACPI_INTERFACE_HANDLER",          SRC_TYPE_SIMPLE},
    {"ACPI_IDENTIFIER_TABLE",           SRC_TYPE_STRUCT},
    {"ACPI_INIT_WALK_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_INTEGER",                    SRC_TYPE_SIMPLE},
    {"ACPI_INTEGER_OVERLAY",            SRC_TYPE_STRUCT},
    {"ACPI_INTEGRITY_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_INTERFACE_INFO",             SRC_TYPE_STRUCT},
    {"ACPI_INTERNAL_METHOD",            SRC_TYPE_SIMPLE},
    {"ACPI_INTERPRETER_MODE",           SRC_TYPE_SIMPLE},
    {"ACPI_IO_ADDRESS",                 SRC_TYPE_SIMPLE},
    {"ACPI_IO_ATTRIBUTE",               SRC_TYPE_STRUCT},
    {"ACPI_MEM_SPACE_CONTEXT",          SRC_TYPE_STRUCT},
    {"ACPI_MEMORY_ATTRIBUTE",           SRC_TYPE_STRUCT},
    {"ACPI_MEMORY_LIST",                SRC_TYPE_STRUCT},
```

```
259     {"ACPI_MTMR_ENTRY",                 SRC_TYPE_STRUCT},
260     {"ACPI_MUTEX",                      SRC_TYPE_SIMPLE},
261     {"ACPI_MUTEX_HANDLE",               SRC_TYPE_SIMPLE},
262     {"ACPI_MUTEX_INFO",                 SRC_TYPE_STRUCT},
263     {"ACPI_NAME",                       SRC_TYPE_SIMPLE},
264     {"ACPI_NAME_INFO",                  SRC_TYPE_STRUCT},
265     {"ACPI_NAME_UNION",                 SRC_TYPE_UNION},
266     {"ACPI_NAMESPACE_NODE",             SRC_TYPE_STRUCT},
267     {"ACPI_NAMESTRING_INFO",            SRC_TYPE_STRUCT},
268     {"ACPI_NATIVE_INT",                 SRC_TYPE_SIMPLE},
269     {"ACPI_NATIVE_UINT",                SRC_TYPE_SIMPLE},
270     {"ACPI_NOTIFY_HANDLER",             SRC_TYPE_SIMPLE},
271     {"ACPI_NOTIFY_INFO",                SRC_TYPE_STRUCT},
272     {"ACPI_NS_SEARCH_DATA",             SRC_TYPE_STRUCT},
273     {"ACPI_OBJ_INFO_HEADER",            SRC_TYPE_STRUCT},
274     {"ACPI_OBJECT",                     SRC_TYPE_UNION},
275     {"ACPI_OBJECT_ADDR_HANDLER",        SRC_TYPE_STRUCT},
276     {"ACPI_OBJECT_BANK_FIELD",          SRC_TYPE_STRUCT},
277     {"ACPI_OBJECT_BUFFER",              SRC_TYPE_STRUCT},
278     {"ACPI_OBJECT_BUFFER_FIELD",        SRC_TYPE_STRUCT},
279     {"ACPI_OBJECT_CACHE_LIST",          SRC_TYPE_STRUCT},
280     {"ACPI_OBJECT_COMMON",              SRC_TYPE_STRUCT},
281     {"ACPI_OBJECT_CONVERTER",           SRC_TYPE_SIMPLE},
282     {"ACPI_OBJECT_DATA",                SRC_TYPE_STRUCT},
283     {"ACPI_OBJECT_DEVICE",              SRC_TYPE_STRUCT},
284     {"ACPI_OBJECT_EVENT",               SRC_TYPE_STRUCT},
285     {"ACPI_OBJECT_EXTRA",               SRC_TYPE_STRUCT},
286     {"ACPI_OBJECT_FIELD_COMMON",        SRC_TYPE_STRUCT},
287     {"ACPI_OBJECT_HANDLER",             SRC_TYPE_SIMPLE},
288     {"ACPI_OBJECT_INDEX_FIELD",         SRC_TYPE_STRUCT},
289     {"ACPI_OBJECT_INTEGER",             SRC_TYPE_STRUCT},
290     {"ACPI_OBJECT_LIST",                SRC_TYPE_STRUCT},
291     {"ACPI_OBJECT_METHOD",              SRC_TYPE_STRUCT},
292     {"ACPI_OBJECT_MUTEX",               SRC_TYPE_STRUCT},
293     {"ACPI_OBJECT_NOTIFY_COMMON",       SRC_TYPE_STRUCT},
294     {"ACPI_OBJECT_NOTIFY_HANDLER",      SRC_TYPE_STRUCT},
295     {"ACPI_OBJECT_PACKAGE",             SRC_TYPE_STRUCT},
296     {"ACPI_OBJECT_POWER_RESOURCE",      SRC_TYPE_STRUCT},
297     {"ACPI_OBJECT_PROCESSOR",           SRC_TYPE_STRUCT},
298     {"ACPI_OBJECT_REFERENCE",           SRC_TYPE_STRUCT},
299     {"ACPI_OBJECT_REGION",              SRC_TYPE_STRUCT},
300     {"ACPI_OBJECT_REGION_FIELD",        SRC_TYPE_STRUCT},
301     {"ACPI_OBJECT_STRING",              SRC_TYPE_STRUCT},
302     {"ACPI_OBJECT_THERMAL_ZONE",        SRC_TYPE_STRUCT},
303     {"ACPI_OBJECT_TYPE",                SRC_TYPE_SIMPLE},
304     {"ACPI_OBJECT_TYPE8",               SRC_TYPE_SIMPLE},
305     {"ACPI_OP_WALK_INFO",               SRC_TYPE_STRUCT},
306     {"ACPI_OPCODE_INFO",                SRC_TYPE_STRUCT},
307     {"ACPI_OPERAND_OBJECT",             SRC_TYPE_UNION},
308     {"ACPI_OSD_HANDLER",                SRC_TYPE_SIMPLE},
309     {"ACPI_OSD_EXEC_CALLBACK",          SRC_TYPE_SIMPLE},
310     {"ACPI_OWNER_ID",                   SRC_TYPE_SIMPLE},
311     {"ACPI_PACKAGE_INFO",               SRC_TYPE_STRUCT},
312     {"ACPI_PACKAGE_INFO2",              SRC_TYPE_STRUCT},
313     {"ACPI_PACKAGE_INFO3",              SRC_TYPE_STRUCT},
314     {"ACPI_PARSE_DOWNWARDS",            SRC_TYPE_SIMPLE},
315     {"ACPI_PARSE_OBJ_ASL",              SRC_TYPE_STRUCT},
316     {"ACPI_PARSE_OBJ_COMMON",           SRC_TYPE_STRUCT},
317     {"ACPI_PARSE_OBJ_NAMED",            SRC_TYPE_STRUCT},
318     {"ACPI_PARSE_OBJECT",               SRC_TYPE_UNION},
319     {"ACPI_PARSE_STATE",                SRC_TYPE_STRUCT},
320     {"ACPI_PARSE_UPWARDS",              SRC_TYPE_SIMPLE},
321     {"ACPI_PARSE_VALUE",                SRC_TYPE_UNION},
322     {"ACPI_PCI_DEVICE",                 SRC_TYPE_STRUCT},
323     {"ACPI_PCI_ID",                     SRC_TYPE_STRUCT},
324     {"ACPI_PCI_ROUTING_TABLE",          SRC_TYPE_STRUCT},
```



```
325     {"ACPI_PHYSICAL_ADDRESS",           SRC_TYPE_SIMPLE},
326     {"ACPI_PKG_CALLBACK",               SRC_TYPE_SIMPLE},
327     {"ACPI_PKG_INFO",                   SRC_TYPE_STRUCT},
328     {"ACPI_PKG_STATE",                  SRC_TYPE_STRUCT},
329     {"ACPI_PMTT_HEADER",                SRC_TYPE_STRUCT},
330     {"ACPI_PNP_DEVICE_ID",              SRC_TYPE_STRUCT},
331     {"ACPI_PNP_DEVICE_ID_LIST",         SRC_TYPE_STRUCT},
332     {"ACPI_POINTER",                    SRC_TYPE_STRUCT},
333     {"ACPI_POINTERS",                   SRC_TYPE_UNION},
334     {"ACPI_PORT_INFO",                  SRC_TYPE_STRUCT},
335     {"ACPI_PREDEFINED_DATA",            SRC_TYPE_STRUCT},
336     {"ACPI_PREDEFINED_INFO",            SRC_TYPE_UNION},
337     {"ACPI_PREDEFINED_NAMES",           SRC_TYPE_STRUCT},
338     {"ACPI_PSCOPE_STATE",               SRC_TYPE_STRUCT},
339     {"ACPI_RASF_PARAMETER_BLOCK",       SRC_TYPE_STRUCT},
340     {"ACPI_RASF_PATROL_SCRUB_PARAMETER", SRC_TYPE_STRUCT},
341     {"ACPI_RASF_SHARED_MEMORY",         SRC_TYPE_STRUCT},
342     {"ACPI_REPAIR_FUNCTION",            SRC_TYPE_SIMPLE},
343     {"ACPI_REPAIR_INFO",                SRC_TYPE_STRUCT},
344     {"ACPI_RESOURCE",                   SRC_TYPE_STRUCT},
345     {"ACPI_RESOURCE_HANDLER",           SRC_TYPE_SIMPLE},
346     {"ACPI_RESOURCE_ADDRESS",           SRC_TYPE_STRUCT},
347     {"ACPI_RESOURCE_ADDRESS16",         SRC_TYPE_STRUCT},
348     {"ACPI_RESOURCE_ADDRESS32",         SRC_TYPE_STRUCT},
349     {"ACPI_RESOURCE_ADDRESS64",         SRC_TYPE_STRUCT},
350     {"ACPI_RESOURCE_COMMON_SERIALBUS",  SRC_TYPE_STRUCT},
351     {"ACPI_RESOURCE_EXTENDED_ADDRESS64", SRC_TYPE_STRUCT},
352     {"ACPI_RESOURCE_ATTRIBUTE",         SRC_TYPE_UNION},
353     {"ACPI_RESOURCE_DATA",              SRC_TYPE_UNION},
354     {"ACPI_RESOURCE_DMA",               SRC_TYPE_STRUCT},
355     {"ACPI_RESOURCE_END_TAG",           SRC_TYPE_STRUCT},
356     {"ACPI_RESOURCE_EXTENDED_IRQ",      SRC_TYPE_STRUCT},
357     {"ACPI_RESOURCE_FIXED_DMA",         SRC_TYPE_STRUCT},
358     {"ACPI_RESOURCE_FIXED_IO",          SRC_TYPE_STRUCT},
359     {"ACPI_RESOURCE_FIXED_MEMORY32",    SRC_TYPE_STRUCT},
360     {"ACPI_RESOURCE_GENERIC_REGISTER",  SRC_TYPE_STRUCT},
361     {"ACPI_RESOURCE_GPIO",              SRC_TYPE_STRUCT},
362     {"ACPI_RESOURCE_I2C_SERIALBUS",     SRC_TYPE_STRUCT},
363     {"ACPI_RESOURCE_INFO",              SRC_TYPE_STRUCT},
364     {"ACPI_RESOURCE_IO",                SRC_TYPE_STRUCT},
365     {"ACPI_RESOURCE_IRQ",               SRC_TYPE_STRUCT},
366     {"ACPI_RESOURCE_MEMORY24",          SRC_TYPE_STRUCT},
367     {"ACPI_RESOURCE_MEMORY32",          SRC_TYPE_STRUCT},
368     {"ACPI_RESOURCE_SOURCE",            SRC_TYPE_STRUCT},
369     {"ACPI_RESOURCE_SPI_SERIALBUS",     SRC_TYPE_STRUCT},
370     {"ACPI_RESOURCE_START_DEPENDENT",   SRC_TYPE_STRUCT},
371     {"ACPI_RESOURCE_TAG",               SRC_TYPE_STRUCT},
372     {"ACPI_RESOURCE_TYPE",              SRC_TYPE_SIMPLE},
373     {"ACPI_RESOURCE_UART_SERIALBUS",    SRC_TYPE_STRUCT},
374     {"ACPI_RESOURCE_VENDOR",            SRC_TYPE_STRUCT},
375     {"ACPI_RESOURCE_VENDOR_TYPED",      SRC_TYPE_STRUCT},
376     {"ACPI_RESULT_VALUES",              SRC_TYPE_STRUCT},
377     {"ACPI_ROUND_UP_TO_32_BIT",         SRC_TYPE_SIMPLE},
378     {"ACPI_RSCONVERT_INFO",             SRC_TYPE_STRUCT},
379     {"ACPI_RSDUMP_INFO",                SRC_TYPE_STRUCT},
380     {"ACPI_RW_LOCK",                    SRC_TYPE_STRUCT},
381     {"ACPI_S3PT_HEADER",                SRC_TYPE_STRUCT},
382     {"ACPI_SCI_HANDLER",                SRC_TYPE_SIMPLE},
383     {"ACPI_SCI_HANDLER_INFO",           SRC_TYPE_STRUCT},
384     {"ACPI_SCOPE_STATE",                SRC_TYPE_STRUCT},
385     {"ACPI_SEMAPHORE",                  SRC_TYPE_SIMPLE},
386     {"ACPI_SIGNAL_FATAL_INFO",          SRC_TYPE_STRUCT},
387     {"ACPI_SIMPLE_REPAIR_INFO",         SRC_TYPE_STRUCT},
388     {"ACPI_SIZE",                       SRC_TYPE_SIMPLE},
389     {"ACPI_SLEEP_FUNCTION",             SRC_TYPE_SIMPLE},
390     {"ACPI_SLEEP_FUNCTIONS",            SRC_TYPE_STRUCT},
```

```
391     {"ACPI_SPINLOCK",                       SRC_TYPE_SIMPLE},
392     {"ACPI_STATISTICS",                     SRC_TYPE_STRUCT},
393     {"ACPI_STATUS",                         SRC_TYPE_SIMPLE},
394     {"ACPI_STRING",                         SRC_TYPE_SIMPLE},
395     {"ACPI_STRING_TABLE",                   SRC_TYPE_STRUCT},
396     {"ACPI_SUBTABLE_HEADER",                SRC_TYPE_STRUCT},
397     {"ACPI_SYSTEM_INFO",                    SRC_TYPE_STRUCT},
398     {"ACPI_TABLE_DESC",                     SRC_TYPE_STRUCT},
399     {"ACPI_TABLE_HANDLER",                  SRC_TYPE_SIMPLE},
400     {"ACPI_TABLE_HEADER",                   SRC_TYPE_STRUCT},
401     {"ACPI_TABLE_INFO",                     SRC_TYPE_STRUCT},
402     {"ACPI_TABLE_LIST",                     SRC_TYPE_STRUCT},
403     {"ACPI_TABLE_MTMR",                     SRC_TYPE_STRUCT},
404     {"ACPI_TABLE_SUPPORT",                  SRC_TYPE_STRUCT},
405     {"ACPI_TABLE_TYPE",                     SRC_TYPE_SIMPLE},
406     {"ACPI_TABLE_VRTC",                     SRC_TYPE_STRUCT},
407     {"ACPI_TAG_INFO",                       SRC_TYPE_STRUCT},
408     {"ACPI_THREAD_ID",                      SRC_TYPE_SIMPLE},
409     {"ACPI_THREAD_STATE",                   SRC_TYPE_STRUCT},
410     {"ACPI_TYPED_IDENTIFIER_TABLE",         SRC_TYPE_STRUCT},
411     {"ACPI_UINTPTR_T",                      SRC_TYPE_SIMPLE},
412     {"ACPI_UPDATE_STATE",                   SRC_TYPE_STRUCT},
413     {"ACPI_UUID",                           SRC_TYPE_STRUCT},
414     {"ACPI_VENDOR_UUID",                    SRC_TYPE_STRUCT},
415     {"ACPI_VENDOR_WALK_INFO",               SRC_TYPE_STRUCT},
416     {"ACPI_VRTC_ENTRY",                     SRC_TYPE_STRUCT},
417     {"ACPI_WALK_AML_CALLBACK",              SRC_TYPE_SIMPLE},
418     {"ACPI_WALK_CALLBACK",                  SRC_TYPE_SIMPLE},
419     {"ACPI_WALK_RESOURCE_CALLBACK",         SRC_TYPE_SIMPLE},
420     {"ACPI_WALK_INFO",                      SRC_TYPE_STRUCT},
421     {"ACPI_WALK_STATE",                     SRC_TYPE_STRUCT},
422     {"ACPI_WHEA_HEADER",                    SRC_TYPE_STRUCT},

424     /* Buffers related to predefined ACPI names (_PLD, etc.) */

426     {"ACPI_FDE_INFO",                       SRC_TYPE_STRUCT},
427     {"ACPI_GRT_INFO",                       SRC_TYPE_STRUCT},
428     {"ACPI_GTM_INFO",                       SRC_TYPE_STRUCT},
429     {"ACPI_PLD_INFO",                       SRC_TYPE_STRUCT},

431     /* Resources */

433     {"ACPI_RS_LENGTH",                      SRC_TYPE_SIMPLE},
434     {"ACPI_RSDESC_SIZE",                    SRC_TYPE_SIMPLE},

436     {"AML_RESOURCE",                        SRC_TYPE_UNION},
437     {"AML_RESOURCE_ADDRESS",                SRC_TYPE_STRUCT},
438     {"AML_RESOURCE_ADDRESS16",              SRC_TYPE_STRUCT},
439     {"AML_RESOURCE_ADDRESS32",              SRC_TYPE_STRUCT},
440     {"AML_RESOURCE_ADDRESS64",              SRC_TYPE_STRUCT},
441     {"AML_RESOURCE_COMMON_SERIALBUS",       SRC_TYPE_STRUCT},
442     {"AML_RESOURCE_DMA",                    SRC_TYPE_STRUCT},
443     {"AML_RESOURCE_END_DEPENDENT",          SRC_TYPE_STRUCT},
444     {"AML_RESOURCE_END_TAG",                SRC_TYPE_STRUCT},
445     {"AML_RESOURCE_EXTENDED_ADDRESS64",     SRC_TYPE_STRUCT},
446     {"AML_RESOURCE_EXTENDED_IRQ",           SRC_TYPE_STRUCT},
447     {"AML_RESOURCE_FIXED_DMA",              SRC_TYPE_STRUCT},
448     {"AML_RESOURCE_FIXED_IO",               SRC_TYPE_STRUCT},
449     {"AML_RESOURCE_FIXED_MEMORY32",         SRC_TYPE_STRUCT},
450     {"AML_RESOURCE_GENERIC_REGISTER",       SRC_TYPE_STRUCT},
451     {"AML_RESOURCE_GPIO",                   SRC_TYPE_STRUCT},
452     {"AML_RESOURCE_IO",                     SRC_TYPE_STRUCT},
453     {"AML_RESOURCE_I2C_SERIALBUS",          SRC_TYPE_STRUCT},
454     {"AML_RESOURCE_IRQ",                    SRC_TYPE_STRUCT},
455     {"AML_RESOURCE_IRQ_NOFLAGS",            SRC_TYPE_STRUCT},
456     {"AML_RESOURCE_LARGE_HEADER",           SRC_TYPE_STRUCT},
```



```
457     {"AML_RESOURCE_MEMORY24",               SRC_TYPE_STRUCT},
458     {"AML_RESOURCE_MEMORY32",               SRC_TYPE_STRUCT},
459     {"AML_RESOURCE_SMALL_HEADER",           SRC_TYPE_STRUCT},
460     {"AML_RESOURCE_SPI_SERIALBUS",          SRC_TYPE_STRUCT},
461     {"AML_RESOURCE_START_DEPENDENT",        SRC_TYPE_STRUCT},
462     {"AML_RESOURCE_START_DEPENDENT_NOPRIO", SRC_TYPE_STRUCT},
463     {"AML_RESOURCE_UART_SERIALBUS",         SRC_TYPE_STRUCT},
464     {"AML_RESOURCE_VENDOR_LARGE",           SRC_TYPE_STRUCT},
465     {"AML_RESOURCE_VENDOR_SMALL",           SRC_TYPE_STRUCT},

467     {"APIC_HEADER",                         SRC_TYPE_STRUCT},
468     {"AE_DEBUG_REGIONS",                    SRC_TYPE_STRUCT},
469     {"AE_REGION",                           SRC_TYPE_STRUCT},
470     {"AE_TABLE_DESC",                       SRC_TYPE_STRUCT},
471     {"ASL_ANALYSIS_WALK_INFO",              SRC_TYPE_STRUCT},
472     {"ASL_ERROR_MSG",                       SRC_TYPE_STRUCT},
473     {"ASL_EVENT_INFO",                      SRC_TYPE_STRUCT},
474     {"ASL_FILE_INFO",                       SRC_TYPE_STRUCT},
475     {"ASL_FILE_STATUS",                     SRC_TYPE_STRUCT},
476     {"ASL_INCLUDE_DIR",                     SRC_TYPE_STRUCT},
477     {"ASL_LISTING_NODE",                    SRC_TYPE_STRUCT},
478     {"ASL_MAPPING_ENTRY",                   SRC_TYPE_STRUCT},
479     {"ASL_METHOD_INFO",                     SRC_TYPE_STRUCT},
480     {"ASL_RESERVED_INFO",                   SRC_TYPE_STRUCT},
481     {"ASL_RESOURCE_NODE",                   SRC_TYPE_STRUCT},
482     {"ASL_WALK_CALLBACK",                   SRC_TYPE_SIMPLE},
483     {"UINT64_OVERLAY",                      SRC_TYPE_UNION},
484     {"UINT64_STRUCT",                       SRC_TYPE_STRUCT},

486     /*
487      * Acpi table definition names.
488      */
489     {"ACPI_TABLE_ASF",                      SRC_TYPE_STRUCT},
490     {"ACPI_TABLE_BERT",                     SRC_TYPE_STRUCT},
491     {"ACPI_TABLE_BGRT",                     SRC_TYPE_STRUCT},
492     {"ACPI_TABLE_BOOT",                     SRC_TYPE_STRUCT},
493     {"ACPI_TABLE_CPEP",                     SRC_TYPE_STRUCT},
494     {"ACPI_TABLE_CSRT",                     SRC_TYPE_STRUCT},
495     {"ACPI_TABLE_DBG2",                     SRC_TYPE_STRUCT},
496     {"ACPI_TABLE_DBGP",                     SRC_TYPE_STRUCT},
497     {"ACPI_TABLE_DMAR",                     SRC_TYPE_STRUCT},
498     {"ACPI_TABLE_DRTM",                     SRC_TYPE_STRUCT},
499     {"ACPI_TABLE_ECDT",                     SRC_TYPE_STRUCT},
500     {"ACPI_TABLE_EINJ",                     SRC_TYPE_STRUCT},
501     {"ACPI_TABLE_ERST",                     SRC_TYPE_STRUCT},
502     {"ACPI_TABLE_FACS",                     SRC_TYPE_STRUCT},
503     {"ACPI_TABLE_FADT",                     SRC_TYPE_STRUCT},
504     {"ACPI_TABLE_FPDT",                     SRC_TYPE_STRUCT},
505     {"ACPI_TABLE_HEST",                     SRC_TYPE_STRUCT},
506     {"ACPI_TABLE_HPET",                     SRC_TYPE_STRUCT},
507     {"ACPI_TABLE_IBFT",                     SRC_TYPE_STRUCT},
508     {"ACPI_TABLE_IVRS",                     SRC_TYPE_STRUCT},
509     {"ACPI_TABLE_MADT",                     SRC_TYPE_STRUCT},
510     {"ACPI_TABLE_MCFG",                     SRC_TYPE_STRUCT},
511     {"ACPI_TABLE_MCHI",                     SRC_TYPE_STRUCT},
512     {"ACPI_TABLE_MPST",                     SRC_TYPE_STRUCT},
513     {"ACPI_TABLE_MSCT",                     SRC_TYPE_STRUCT},
514     {"ACPI_TABLE_PCCT",                     SRC_TYPE_STRUCT},
515     {"ACPI_TABLE_RSDP",                     SRC_TYPE_STRUCT},
516     {"ACPI_TABLE_RSDT",                     SRC_TYPE_STRUCT},
517     {"ACPI_TABLE_MCHI",                     SRC_TYPE_STRUCT},
518     {"ACPI_TABLE_S3PT",                     SRC_TYPE_STRUCT},
519     {"ACPI_TABLE_SBST",                     SRC_TYPE_STRUCT},
520     {"ACPI_TABLE_SLIC",                     SRC_TYPE_STRUCT},
521     {"ACPI_TABLE_SLIT",                     SRC_TYPE_STRUCT},
522     {"ACPI_TABLE_SPCR",                     SRC_TYPE_STRUCT},
```

```
523     {"ACPI_TABLE_SPMI",                 SRC_TYPE_STRUCT},
524     {"ACPI_TABLE_SRAT",                 SRC_TYPE_STRUCT},
525     {"ACPI_TABLE_TCPA",                 SRC_TYPE_STRUCT},
526     {"ACPI_TABLE_TPM2",                 SRC_TYPE_STRUCT},
527     {"ACPI_TABLE_UEFI",                 SRC_TYPE_STRUCT},
528     {"ACPI_TABLE_WAET",                 SRC_TYPE_STRUCT},
529     {"ACPI_TABLE_WDAT",                 SRC_TYPE_STRUCT},
530     {"ACPI_TABLE_WDDT",                 SRC_TYPE_STRUCT},
531     {"ACPI_TABLE_WDRT",                 SRC_TYPE_STRUCT},
532     {"ACPI_TABLE_XSDT",                 SRC_TYPE_STRUCT},

534     {"ACPI_ASF_ADDRESS",                SRC_TYPE_STRUCT},
535     {"ACPI_ASF_ALERT",                  SRC_TYPE_STRUCT},
536     {"ACPI_ASF_ALERT_DATA",             SRC_TYPE_STRUCT},
537     {"ACPI_ASF_CONTROL_DATA",           SRC_TYPE_STRUCT},
538     {"ACPI_ASF_HEADER",                 SRC_TYPE_STRUCT},
539     {"ACPI_ASF_INFO",                   SRC_TYPE_STRUCT},
540     {"ACPI_ASF_REMOTE",                 SRC_TYPE_STRUCT},
541     {"ACPI_ASF_RMCP",                   SRC_TYPE_STRUCT},
542     {"ACPI_BERT_REGION",                SRC_TYPE_STRUCT},
543     {"ACPI_CPEP_POLLING",               SRC_TYPE_STRUCT},
544     {"ACPI_CSRT_GROUP",                 SRC_TYPE_STRUCT},
545     {"ACPI_CSRT_DESCRIPTOR",            SRC_TYPE_STRUCT},
546     {"ACPI_CSRT_SHARED_INFO",           SRC_TYPE_STRUCT},
547     {"ACPI_DBG2_DEVICE",                SRC_TYPE_STRUCT},
548     {"ACPI_DMAR_HEADER",                SRC_TYPE_STRUCT},
549     {"ACPI_DMAR_DEVICE_SCOPE",          SRC_TYPE_STRUCT},
550     {"ACPI_DMAR_ATSR",                  SRC_TYPE_STRUCT},
551     {"ACPI_DMAR_RHSA",                  SRC_TYPE_STRUCT},
552     {"ACPI_DMAR_HARDWARE_UNIT",         SRC_TYPE_STRUCT},
553     {"ACPI_DMAR_RESERVED_MEMORY",       SRC_TYPE_STRUCT},
554     {"ACPI_EINJ_ENTRY",                 SRC_TYPE_STRUCT},
555     {"ACPI_EINJ_TRIGGER",               SRC_TYPE_STRUCT},
556     {"ACPI_FPDT_HEADER",                SRC_TYPE_STRUCT},
557     {"ACPI_FPDT_BOOT",                  SRC_TYPE_STRUCT},
558     {"ACPI_FPDT_S3PT_PTR",              SRC_TYPE_STRUCT},
559     {"ACPI_ERST_ENTRY",                 SRC_TYPE_STRUCT},
560     {"ACPI_ERST_INFO",                  SRC_TYPE_STRUCT},
561     {"ACPI_HEST_AER_COMMON",            SRC_TYPE_STRUCT},
562     {"ACPI_HEST_HEADER",                SRC_TYPE_STRUCT},
563     {"ACPI_HEST_NOTIFY",                SRC_TYPE_STRUCT},
564     {"ACPI_HEST_IA_ERROR_BANK",         SRC_TYPE_STRUCT},
565     {"ACPI_HEST_IA_MACHINE_CHECK",      SRC_TYPE_STRUCT},
566     {"ACPI_HEST_IA_CORRECTED",          SRC_TYPE_STRUCT},
567     {"ACPI_HEST_IA_NMI",                SRC_TYPE_STRUCT},
568     {"ACPI_HEST_AER_ROOT",              SRC_TYPE_STRUCT},
569     {"ACPI_HEST_AER",                   SRC_TYPE_STRUCT},
570     {"ACPI_HEST_AER_BRIDGE",            SRC_TYPE_STRUCT},
571     {"ACPI_HEST_GENERIC",               SRC_TYPE_STRUCT},
572     {"ACPI_HEST_GENERIC_STATUS",        SRC_TYPE_STRUCT},
573     {"ACPI_HEST_GENERIC_DATA",          SRC_TYPE_STRUCT},
574     {"ACPI_IBFT_HEADER",                SRC_TYPE_STRUCT},
575     {"ACPI_IBFT_CONTROL",               SRC_TYPE_STRUCT},
576     {"ACPI_IBFT_INITIATOR",             SRC_TYPE_STRUCT},
577     {"ACPI_IBFT_NIC",                   SRC_TYPE_STRUCT},
578     {"ACPI_IBFT_TARGET",                SRC_TYPE_STRUCT},
579     {"ACPI_IVRS_HEADER",                SRC_TYPE_STRUCT},
580     {"ACPI_IVRS_HARDWARE",              SRC_TYPE_STRUCT},
581     {"ACPI_IVRS_DE_HEADER",             SRC_TYPE_STRUCT},
582     {"ACPI_IVRS_DEVICE4",               SRC_TYPE_STRUCT},
583     {"ACPI_IVRS_DEVICE8A",              SRC_TYPE_STRUCT},
584     {"ACPI_IVRS_DEVICE8B",              SRC_TYPE_STRUCT},
585     {"ACPI_IVRS_DEVICE8C",              SRC_TYPE_STRUCT},
586     {"ACPI_IVRS_MEMORY",                SRC_TYPE_STRUCT},
587     {"ACPI_MADT_ADDRESS_OVERRIDE",      SRC_TYPE_STRUCT},
588     {"ACPI_MADT_HEADER",                SRC_TYPE_STRUCT},
```



```
589     {"ACPI_MADT_IO_APIC",               SRC_TYPE_STRUCT},
590     {"ACPI_MADT_IO_SAPIC",              SRC_TYPE_STRUCT},
591     {"ACPI_MADT_LOCAL_APIC",            SRC_TYPE_STRUCT},
592     {"ACPI_MADT_LOCAL_APIC_NMI",        SRC_TYPE_STRUCT},
593     {"ACPI_MADT_LOCAL_APIC_OVERRIDE",   SRC_TYPE_STRUCT},
594     {"ACPI_MADT_LOCAL_SAPIC",           SRC_TYPE_STRUCT},
595     {"ACPI_MADT_LOCAL_X2APIC",          SRC_TYPE_STRUCT},
596     {"ACPI_MADT_LOCAL_X2APIC_NMI",      SRC_TYPE_STRUCT},
597     {"ACPI_MADT_GENERIC_DISTRIBUTOR",   SRC_TYPE_STRUCT},
598     {"ACPI_MADT_GENERIC_INTERRUPT",     SRC_TYPE_STRUCT},
599     {"ACPI_MADT_INTERRUPT_OVERRIDE",    SRC_TYPE_STRUCT},
600     {"ACPI_MADT_INTERRUPT_SOURCE",      SRC_TYPE_STRUCT},
601     {"ACPI_MADT_NMI_SOURCE",            SRC_TYPE_STRUCT},
602     {"ACPI_MADT_PROCESSOR_APIC",        SRC_TYPE_STRUCT},
603     {"ACPI_MPST_COMPONENT",             SRC_TYPE_STRUCT},
604     {"ACPI_MPST_DATA_HDR",              SRC_TYPE_STRUCT},
605     {"ACPI_MPST_POWER_DATA",            SRC_TYPE_STRUCT},
606     {"ACPI_MPST_POWER_NODE",            SRC_TYPE_STRUCT},
607     {"ACPI_MPST_POWER_STATE",           SRC_TYPE_STRUCT},
608     {"ACPI_MCFG_ALLOCATION",            SRC_TYPE_STRUCT},
609     {"ACPI_MSCT_PROXIMITY",             SRC_TYPE_STRUCT},
610     {"ACPI_PCCT_SUBSPACE",              SRC_TYPE_STRUCT},
611     {"ACPI_RSDP_COMMON",                SRC_TYPE_STRUCT},
612     {"ACPI_RSDP_EXTENSION",             SRC_TYPE_STRUCT},
613     {"ACPI_S3PT_RESUME",                SRC_TYPE_STRUCT},
614     {"ACPI_S3PT_SUSPEND",               SRC_TYPE_STRUCT},
615     {"ACPI_SLIC_HEADER",                SRC_TYPE_STRUCT},
616     {"ACPI_SLIC_KEY",                   SRC_TYPE_STRUCT},
617     {"ACPI_SLIC_MARKER",                SRC_TYPE_STRUCT},
618     {"ACPI_SRAT_CPU_AFFINITY",          SRC_TYPE_STRUCT},
619     {"ACPI_SRAT_HEADER",                SRC_TYPE_STRUCT},
620     {"ACPI_SRAT_MEM_AFFINITY",          SRC_TYPE_STRUCT},
621     {"ACPI_SRAT_X2APIC_CPU_AFFINITY",   SRC_TYPE_STRUCT},
622     {"ACPI_TPM2_CONTROL",               SRC_TYPE_STRUCT},
623     {"ACPI_WDAT_ENTRY",                 SRC_TYPE_STRUCT},

625     /* Data Table compiler */

627     {"DT_FIELD",                        SRC_TYPE_STRUCT},
628     {"DT_SUBTABLE",                     SRC_TYPE_STRUCT},
629     {"DT_WALK_CALLBACK",                SRC_TYPE_SIMPLE},

631     /* iASL preprocessor */

633     {"PR_DEFINE_INFO",                  SRC_TYPE_STRUCT},
634     {"PR_DIRECTIVE_INFO",               SRC_TYPE_STRUCT},
635     {"PR_FILE_NODE",                    SRC_TYPE_STRUCT},
636     {"PR_LINE_MAPPING",                 SRC_TYPE_STRUCT},
637     {"PR_MACRO_ARG",                    SRC_TYPE_STRUCT},
638     {"PR_OPERATOR_INFO",                SRC_TYPE_STRUCT},

640     /* AcpiHelp utility */

642     {"AH_AML_OPCODE",                   SRC_TYPE_STRUCT},
643     {"AH_ASL_OPERATOR",                 SRC_TYPE_STRUCT},
644     {"AH_ASL_KEYWORD",                  SRC_TYPE_STRUCT},
645     {"AH_PREDEFINED_NAME",              SRC_TYPE_STRUCT},

647     /* AcpiXtract utility */

649     {"AX_TABLE_INFO",                   SRC_TYPE_STRUCT},

651     {NULL, 0}
652 };
```

```
655 ACPI_IDENTIFIER_TABLE       LinuxAddStruct[] = {
656     {"acpi_namespace_node"},
657     {"acpi_parse_object"},
658     {"acpi_table_desc"},
659     {"acpi_walk_state"},
660     {NULL}
661 };


664 ACPI_IDENTIFIER_TABLE       LinuxEliminateMacros[] = {

666     {"ACPI_GET_ADDRESS"},
667     {"ACPI_VALID_ADDRESS"},
668     {NULL}
669 };


672 ACPI_IDENTIFIER_TABLE       LinuxEliminateLines_C[] = {

674     {"#define __"},
675     {NULL}
676 };


679 ACPI_IDENTIFIER_TABLE       LinuxEliminateLines_H[] = {

681     {NULL}
682 };


685 ACPI_IDENTIFIER_TABLE       LinuxConditionalIdentifiers[] = {

687 /*    {"ACPI_USE_STANDARD_HEADERS"}, */
688     {"WIN32"},
689     {"_MSC_VER"},
690     {NULL}
691 };


694 ACPI_STRING_TABLE           LinuxSpecialStrings[] = {

696     /* Include file paths */

698     {"\"acpi.h\"",              "<acpi/acpi.h>",               REPLACE_WHOLE_WO
699     {"\"acpiosxf.h\"",          "<acpi/acpiosxf.h>",           REPLACE_WHOLE_WO
700     {"\"acpixf.h\"",            "<acpi/acpixf.h>",             REPLACE_WHOLE_WO
701     {"\"acbuffer.h\"",          "<acpi/acbuffer.h>",           REPLACE_WHOLE_WO
702     {"\"acconfig.h\"",          "<acpi/acconfig.h>",           REPLACE_WHOLE_WO
703     {"\"acexcep.h\"",           "<acpi/acexcep.h>",            REPLACE_WHOLE_WO
704     {"\"acnames.h\"",           "<acpi/acnames.h>",            REPLACE_WHOLE_WO
705     {"\"acoutput.h\"",          "<acpi/acoutput.h>",           REPLACE_WHOLE_WO
706     {"\"acrestyp.h\"",          "<acpi/acrestyp.h>",           REPLACE_WHOLE_WO
707     {"\"actbl.h\"",             "<acpi/actbl.h>",              REPLACE_WHOLE_WO
708     {"\"actbl1.h\"",            "<acpi/actbl1.h>",             REPLACE_WHOLE_WO
709     {"\"actbl2.h\"",            "<acpi/actbl2.h>",             REPLACE_WHOLE_WO
710     {"\"actbl3.h\"",            "<acpi/actbl3.h>",             REPLACE_WHOLE_WO
711     {"\"actypes.h\"",           "<acpi/actypes.h>",            REPLACE_WHOLE_WO
712     {"\"platform/acenv.h\"",    "<acpi/platform/acenv.h>",     REPLACE_WHOLE_WO
713     {"\"acgcc.h\"",             "<acpi/platform/acgcc.h>",     REPLACE_WHOLE_WO
714     {"\"aclinux.h\"",           "<acpi/platform/aclinux.h>",   REPLACE_WHOLE_WO

716     {NULL,                      NULL,              0}
717 };


720 ACPI_IDENTIFIER_TABLE       LinuxSpecialMacros[] = {
```



```
722     {"ACPI_EXPORT_SYMBOL"},
723     {"ACPI_EXPORT_SYMBOL_INIT"},
724     {"ACPI_HW_DEPENDENT_RETURN_OK"},
725     {"ACPI_HW_DEPENDENT_RETURN_STATUS"},
726     {"ACPI_HW_DEPENDENT_RETURN_VOID"},
727     {NULL}
728 };


731 ACPI_CONVERSION_TABLE       LinuxConversionTable = {

733     DualLicenseHeader,
734     FLG_NO_CARRIAGE_RETURNS | FLG_LOWERCASE_DIRNAMES,

736     AcpiIdentifiers,

738     /* C source files */

740     LinuxDataTypes,
741     LinuxEliminateLines_C,
742     NULL,
743     LinuxEliminateMacros,
744     AcpiIdentifiers,
745     NULL,
746     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
747      CVT_CHECK_BRACES | CVT_TRIM_LINES | CVT_BRACES_ON_SAME_LINE |
748      CVT_MIXED_CASE_TO_UNDERSCORES | CVT_LOWER_CASE_IDENTIFIERS |
749      CVT_REMOVE_DEBUG_MACROS | CVT_TRIM_WHITESPACE |
750      CVT_REMOVE_EMPTY_BLOCKS | CVT_SPACES_TO_TABS8),

752     /* C header files */

754     LinuxDataTypes,
755     LinuxEliminateLines_H,
756     LinuxConditionalIdentifiers,
757     NULL,
758     AcpiIdentifiers,
759     NULL,
760     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
761      CVT_TRIM_LINES | CVT_MIXED_CASE_TO_UNDERSCORES |
762      CVT_LOWER_CASE_IDENTIFIERS | CVT_TRIM_WHITESPACE |
763      CVT_REMOVE_EMPTY_BLOCKS| CVT_REDUCE_TYPEDEFS | CVT_SPACES_TO_TABS8),
764 };


767 /******************************************************************************
768  *
769  * Code cleanup translation tables
770  *
771  *****************************************************************************/

773 ACPI_CONVERSION_TABLE       CleanupConversionTable = {

775     NULL,
776     FLG_DEFAULT_FLAGS,
777     NULL,
778     /* C source files */

780     NULL,
781     NULL,
782     NULL,
783     NULL,
784     NULL,
785     NULL,
786     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
```

```
787      CVT_CHECK_BRACES | CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

789     /* C header files */

791     NULL,
792     NULL,
793     NULL,
794     NULL,
795     NULL,
796     NULL,
797     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
798      CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),
799 };


802 ACPI_CONVERSION_TABLE       StatsConversionTable = {

804     NULL,
805     FLG_NO_FILE_OUTPUT,
806     NULL,

808     /* C source files */

810     NULL,
811     NULL,
812     NULL,
813     NULL,
814     NULL,
815     NULL,
816     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
817      CVT_COUNT_SHORTMULTILINE_COMMENTS),

819     /* C header files */

821     NULL,
822     NULL,
823     NULL,
824     NULL,
825     NULL,
826     NULL,
827     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
828      CVT_COUNT_SHORTMULTILINE_COMMENTS),
829 };


832 /******************************************************************************
833  *
834  * Dual License injection translation table
835  *
836  *****************************************************************************/

838 ACPI_CONVERSION_TABLE       LicenseConversionTable = {

840     DualLicenseHeader,
841     FLG_DEFAULT_FLAGS,
842     NULL,

844     /* C source files */

846     NULL,
847     NULL,
848     NULL,
849     NULL,
850     NULL,
851     NULL,
852     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
```



```
853      CVT_COUNT_SHORTMULTILINE_COMMENTS),

855     /* C header files */

857     NULL,
858     NULL,
859     NULL,
860     NULL,
861     NULL,
862     NULL,
863     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
864      CVT_COUNT_SHORTMULTILINE_COMMENTS),
865 };


868 /******************************************************************************
869  *
870  * Customizable translation tables
871  *
872  *****************************************************************************/

874 ACPI_STRING_TABLE           CustomReplacements[] = {


877     {"(c) 1999 - 2012",     "(c) 1999 - 2013",         REPLACE_WHOLE_WORD}, /* M
878     {"(c) 2006 - 2012",     "(c) 2006 - 2013",         REPLACE_WHOLE_WORD}, /* T

880 #if 0
881     {"SUPPORT, ASSISTANCE", "SUPPORT, ASSISTANCE",     REPLACE_WHOLE_WORD}, /* F

883     {"(ACPI_INTEGER)", "(UINT64)",   REPLACE_WHOLE_WORD},
884     {"ACPI_INTEGER        ", "UINT64              ",   REPLACE_WHOLE_WORD},
885     {"ACPI_INTEGER", "UINT64",   REPLACE_WHOLE_WORD},
886     {"ACPI_INTEGER_MAX", "ACPI_UINT64_MAX",   REPLACE_WHOLE_WORD},
887     {"#include \"acpi.h\"",   "#include \"acpi.h\"\n#include \"accommon.h\"",  R
888     {"AcpiTbSumTable", "AcpiTbSumTable",  REPLACE_WHOLE_WORD},
889     {"ACPI_SIG_BOOT", "ACPI_SIG_BOOT",   REPLACE_WHOLE_WORD},
890     {"ACPI_SIG_DBGP", "ACPI_SIG_DBGP",   REPLACE_WHOLE_WORD},
891     {"ACPI_SIG_DSDT", "ACPI_SIG_DSDT",   REPLACE_WHOLE_WORD},
892     {"ACPI_SIG_ECDT", "ACPI_SIG_ECDT",   REPLACE_WHOLE_WORD},
893     {"ACPI_SIG_FACS", "ACPI_SIG_FACS",   REPLACE_WHOLE_WORD},
894     {"ACPI_SIG_FADT", "ACPI_SIG_FADT",   REPLACE_WHOLE_WORD},
895     {"ACPI_SIG_HPET", "ACPI_SIG_HPET",   REPLACE_WHOLE_WORD},
896     {"ACPI_SIG_MADT", "ACPI_SIG_MADT",   REPLACE_WHOLE_WORD},
897     {"ACPI_SIG_MCFG", "ACPI_SIG_MCFG",   REPLACE_WHOLE_WORD},
898     {"ACPI_SIG_PSDT", "ACPI_SIG_PSDT",   REPLACE_WHOLE_WORD},
899     {"ACPI_NAME_RSDP", "ACPI_NAME_RSDP",   REPLACE_WHOLE_WORD},
900     {"ACPI_SIG_RSDP", "ACPI_SIG_RSDP",   REPLACE_WHOLE_WORD},
901     {"ACPI_SIG_RSDT", "ACPI_SIG_RSDT",   REPLACE_WHOLE_WORD},
902     {"ACPI_SIG_SBST", "ACPI_SIG_SBST",   REPLACE_WHOLE_WORD},
903     {"ACPI_SIG_SLIT", "ACPI_SIG_SLIT",   REPLACE_WHOLE_WORD},
904     {"ACPI_SIG_SPCR", "ACPI_SIG_SPCR",   REPLACE_WHOLE_WORD},
905     {"ACPI_SIG_SPIC", "ACPI_SIG_SPIC",   REPLACE_WHOLE_WORD},
906     {"ACPI_SIG_SPMI", "ACPI_SIG_SPMI",   REPLACE_WHOLE_WORD},
907     {"ACPI_SIG_SRAT", "ACPI_SIG_SRAT",   REPLACE_WHOLE_WORD},
908     {"ACPI_SIG_SSDT", "ACPI_SIG_SSDT",   REPLACE_WHOLE_WORD},
909     {"ACPI_SIG_TCPA", "ACPI_SIG_TCPA",   REPLACE_WHOLE_WORD},
910     {"ACPI_SIG_WDRT", "ACPI_SIG_WDRT",   REPLACE_WHOLE_WORD},
911     {"ACPI_SIG_XSDT", "ACPI_SIG_XSDT",   REPLACE_WHOLE_WORD},

913     {"ACPI_ALLOCATE_ZEROED",    "ACPI_ALLOCATE_ZEROED",   REPLACE_WHOLE_WORD},
914     {"ACPI_ALLOCATE",           "ACPI_ALLOCATE",          REPLACE_WHOLE_WORD},
915     {"ACPI_FREE",               "ACPI_FREE",              REPLACE_WHOLE_WORD},

917     "ACPI_NATIVE_UINT",     "ACPI_NATIVE_UINT",         REPLACE_WHOLE_WORD,
918     "ACPI_NATIVE_UINT *",   "ACPI_NATIVE_UINT *",       REPLACE_WHOLE_WORD,
```

```
    "ACPI_NATIVE_UINT",     "ACPI_NATIVE_UINT",         REPLACE_WHOLE_WORD,
    "ACPI_NATIVE_INT",      "ACPI_NATIVE_INT",          REPLACE_WHOLE_WORD,
    "ACPI_NATIVE_INT *",    "ACPI_NATIVE_INT *",        REPLACE_WHOLE_WORD,
    "ACPI_NATIVE_INT",      "ACPI_NATIVE_INT",          REPLACE_WHOLE_WORD,
#endif

    {NULL,                    NULL, 0}
};


ACPI_CONVERSION_TABLE       CustomConversionTable = {

    NULL,
    FLG_DEFAULT_FLAGS,
    NULL,

    /* C source files */

    CustomReplacements,
    LinuxEliminateLines_H,
    NULL,
    NULL,
    NULL,
    NULL,
    (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
     CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

    /* C header files */

    CustomReplacements,
    LinuxEliminateLines_H,
    NULL,
    NULL,
    NULL,
    NULL,
    (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
     CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),
};


/******************************************************************************
 *
 * Indentation result fixup table
 *
 *****************************************************************************/

ACPI_CONVERSION_TABLE       IndentConversionTable = {

    NULL,
    FLG_NO_CARRIAGE_RETURNS,

    NULL,

    /* C source files */

    LinuxSpecialStrings,
    NULL,
    NULL,
    NULL,
    NULL,
    LinuxSpecialMacros,
    (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
     CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

    /* C header files */
```



```
    LinuxSpecialStrings,
    NULL,
    NULL,
    NULL,
    NULL,
    LinuxSpecialMacros,
    (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
     CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),
};
```

```
**********************************************************
    7817 Thu Dec 26 13:50:13 2013
new/usr/src/common/acpica/tools/acpisrc/asutils.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: asutils - common utilities
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpisrc.h"


/******************************************************************************
 *
 * FUNCTION:    AsStrlwr (strlwr)
 *
 * PARAMETERS:  SrcString       - The source string to convert
 *
 * RETURN:      None
 *
 * DESCRIPTION: Convert string to lowercase
 *
 * NOTE: This is not a POSIX function, so it appears here so that we don't have
 * header file issues with the various hosts/compilers/clibs.
 *
 *****************************************************************************/
```



```
void
AsStrlwr (
    char                    *SrcString)
{
    char                    *String;


    /* Walk entire string, lowercasing the letters */

    if (SrcString)
    {
        for (String = SrcString; *String; String++)
        {
            *String = (char) ACPI_TOLOWER (*String);
        }
    }
}


/******************************************************************************
 *
 * FUNCTION:    AsSkipUntilChar
 *
 * DESCRIPTION: Find the next instance of the input character
 *
 *****************************************************************************/

char *
AsSkipUntilChar (
    char                    *Buffer,
    char                    Target)
{

    while (*Buffer != Target)
    {
        if (!*Buffer)
        {
            return (NULL);
        }

        Buffer++;
    }

    return (Buffer);
}


/******************************************************************************
 *
 * FUNCTION:    AsSkipPastChar
 *
 * DESCRIPTION: Find the next instance of the input character, return a buffer
 *              pointer to this character+1.
 *
 *****************************************************************************/

char *
AsSkipPastChar (
    char                    *Buffer,
    char                    Target)
{

    while (*Buffer != Target)
    {
        if (!*Buffer)
        {
```

```
128             return (NULL);
129         }

131         Buffer++;
132     }

134     Buffer++;

136     return (Buffer);
137 }


140 /******************************************************************************
141  *
142  * FUNCTION:    AsReplaceData
143  *
144  * DESCRIPTION: This function inserts and removes data from the file buffer.
145  *              if more data is inserted than is removed, the data in the buffer
146  *              is moved to make room. If less data is inserted than is removed,
147  *              the remaining data is moved to close the hole.
148  *
149  *****************************************************************************/

151 char *
152 AsReplaceData (
153     char                    *Buffer,
154     UINT32                  LengthToRemove,
155     char                    *BufferToAdd,
156     UINT32                  LengthToAdd)
157 {
158     UINT32                  BufferLength;


161     /*
162      * Buffer is a string, so the length must include the terminating zero
163      */
164     BufferLength = strlen (Buffer) + 1;

166     if (LengthToRemove != LengthToAdd)
167     {
168         /*
169          * Move some of the existing data
170          * 1) If adding more bytes than removing, make room for the new data
171          * 2) if removing more bytes than adding, delete the extra space
172          */
173         if (LengthToRemove > 0)
174         {
175             Gbl_MadeChanges = TRUE;
176             memmove ((Buffer + LengthToAdd), (Buffer + LengthToRemove), (BufferL
177         }
178     }

180     /*
181      * Now we can move in the new data
182      */
183     if (LengthToAdd > 0)
184     {
185         Gbl_MadeChanges = TRUE;
186         memmove (Buffer, BufferToAdd, LengthToAdd);
187     }

189     return (Buffer + LengthToAdd);
190 }


193 /******************************************************************************
```



```
194  *
195  * FUNCTION:    AsInsertData
196  *
197  * DESCRIPTION: This function inserts and removes data from the file buffer.
198  *              if more data is inserted than is removed, the data in the buffer
199  *              is moved to make room. If less data is inserted than is removed,
200  *              the remaining data is moved to close the hole.
201  *
202  *****************************************************************************/

204 char *
205 AsInsertData (
206     char                    *Buffer,
207     char                    *BufferToAdd,
208     UINT32                  LengthToAdd)
209 {
210     UINT32                  BufferLength;


213     if (LengthToAdd > 0)
214     {
215         /*
216          * Buffer is a string, so the length must include the terminating zero
217          */
218         BufferLength = strlen (Buffer) + 1;

220         /*
221          * Move some of the existing data
222          * 1) If adding more bytes than removing, make room for the new data
223          * 2) if removing more bytes than adding, delete the extra space
224          */
225         Gbl_MadeChanges = TRUE;
226         memmove ((Buffer + LengthToAdd), Buffer, BufferLength);

228         /*
229          * Now we can move in the new data
230          */
231         memmove (Buffer, BufferToAdd, LengthToAdd);
232     }

234     return (Buffer + LengthToAdd);
235 }


238 /******************************************************************************
239  *
240  * FUNCTION:    AsRemoveData
241  *
242  * DESCRIPTION: This function inserts and removes data from the file buffer.
243  *              if more data is inserted than is removed, the data in the buffer
244  *              is moved to make room. If less data is inserted than is removed,
245  *              the remaining data is moved to close the hole.
246  *
247  *****************************************************************************/

249 char *
250 AsRemoveData (
251     char                    *StartPointer,
252     char                    *EndPointer)
253 {
254     UINT32                  BufferLength;


257     /*
258      * Buffer is a string, so the length must include the terminating zero
259      */
```

```
260     BufferLength = strlen (EndPointer) + 1;

262     Gbl_MadeChanges = TRUE;
263     memmove (StartPointer, EndPointer, BufferLength);

265     return (StartPointer);
266 }
```

```
**********************************************************
   21590 Thu Dec 26 13:50:13 2013
new/usr/src/common/acpica/tools/acpixtract/acpixtract.c
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/******************************************************************************
 *
 * Module Name: acpixtract - convert ascii ACPI tables to binary
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acapps.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

/* Local prototypes */

static void
AxStrlwr (
    char                    *String);

static void
```



```
AxCheckAscii (
    char                    *Name,
    int                     Count);

static void
AxNormalizeSignature (
    char                    *Signature);

static unsigned int
AxGetNextInstance (
    char                    *InputPathname,
    char                    *Signature);

static size_t
AxGetTableHeader (
    FILE                    *InputFile,
    unsigned char           *OutputData);

static unsigned int
AxCountTableInstances (
    char                    *InputPathname,
    char                    *Signature);

int
AxExtractTables (
    char                    *InputPathname,
    char                    *Signature,
    unsigned int            MinimumInstances);

int
AxListTables (
    char                    *InputPathname);

static size_t
AxConvertLine (
    char                    *InputLine,
    unsigned char           *OutputData);

static int
AxIsEmptyLine (
    char                    *Buffer);

typedef struct AxTableInfo
{
    UINT32                  Signature;
    unsigned int            Instances;
    unsigned int            NextInstance;
    struct AxTableInfo      *Next;

} AX_TABLE_INFO;

/* Extraction states */

#define AX_STATE_FIND_HEADER        0
#define AX_STATE_EXTRACT_DATA       1

/* Miscellaneous constants */

#define AX_LINE_BUFFER_SIZE         256
#define AX_MIN_TABLE_NAME_LENGTH    6   /* strlen ("DSDT @") */


static AX_TABLE_INFO        *AxTableListHead = NULL;
static char                 Filename[16];
static unsigned char        Data[16];
static char                 LineBuffer[AX_LINE_BUFFER_SIZE];
```

```
126 static char                HeaderBuffer[AX_LINE_BUFFER_SIZE];
127 static char                InstanceBuffer[AX_LINE_BUFFER_SIZE];


130 /******************************************************************************
131  *
132  * FUNCTION:    AxStrlwr
133  *
134  * PARAMETERS:  String              - Ascii string
135  *
136  * RETURN:      None
137  *
138  * DESCRIPTION: String lowercase function.
139  *
140  *****************************************************************************/

142 static void
143 AxStrlwr (
144     char                    *String)
145 {

147     while (*String)
148     {
149         *String = (char) tolower ((int) *String);
150         String++;
151     }
152 }


155 /******************************************************************************
156  *
157  * FUNCTION:    AxCheckAscii
158  *
159  * PARAMETERS:  Name                - Ascii string, at least as long as Count
160  *              Count               - Number of characters to check
161  *
162  * RETURN:      None
163  *
164  * DESCRIPTION: Ensure that the requested number of characters are printable
165  *              Ascii characters. Sets non-printable and null chars to <space>.
166  *
167  *****************************************************************************/

169 static void
170 AxCheckAscii (
171     char                    *Name,
172     int                     Count)
173 {
174     int                     i;


177     for (i = 0; i < Count; i++)
178     {
179         if (!Name[i] || !isprint ((int) Name[i]))
180         {
181             Name[i] = ' ';
182         }
183     }
184 }


187 /******************************************************************************
188  *
189  * FUNCTION:    AxIsEmptyLine
190  *
191  * PARAMETERS:  Buffer              - Line from input file
```



```
192  *
193  * RETURN:      TRUE if line is empty (zero or more blanks only)
194  *
195  * DESCRIPTION: Determine if an input line is empty.
196  *
197  *****************************************************************************/

199 static int
200 AxIsEmptyLine (
201     char                    *Buffer)
202 {

204     /* Skip all spaces */

206     while (*Buffer == ' ')
207     {
208         Buffer++;
209     }

211     /* If end-of-line, this line is empty */

213     if (*Buffer == '\n')
214     {
215         return (1);
216     }

218     return (0);
219 }


222 /******************************************************************************
223  *
224  * FUNCTION:    AxNormalizeSignature
225  *
226  * PARAMETERS:  Name                - Ascii string containing an ACPI signature
227  *
228  * RETURN:      None
229  *
230  * DESCRIPTION: Change "RSD PTR" to "RSDP"
231  *
232  *****************************************************************************/

234 static void
235 AxNormalizeSignature (
236     char                    *Signature)
237 {

239     if (!strncmp (Signature, "RSD ", 4))
240     {
241         Signature[3] = 'P';
242     }
243 }


246 /******************************************************************************
247  *
248  * FUNCTION:    AxConvertLine
249  *
250  * PARAMETERS:  InputLine           - One line from the input acpidump file
251  *              OutputData          - Where the converted data is returned
252  *
253  * RETURN:      The number of bytes actually converted
254  *
255  * DESCRIPTION: Convert one line of ascii text binary (up to 16 bytes)
256  *
257  *****************************************************************************/
```

```
259 static size_t
260 AxConvertLine (
261     char                    *InputLine,
262     unsigned char           *OutputData)
263 {
264     char                    *End;
265     int                     BytesConverted;
266     int                     Converted[16];
267     int                     i;


270     /* Terminate the input line at the end of the actual data (for sscanf) */

272     End = strstr (InputLine + 2, "  ");
273     if (!End)
274     {
275         return (0); /* Don't understand the format */
276     }
277     *End = 0;

279     /*
280      * Convert one line of table data, of the form:
281      * <offset>: <up to 16 bytes of hex data> <ASCII representation> <newline>
282      *
283      * Example:
284      * 02C0: 5F 53 42 5F 4C 4E 4B 44 00 12 13 04 0C FF FF 08  _SB_LNKD........
285      */
286     BytesConverted = sscanf (InputLine,
287         "%*s %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x",
288         &Converted[0],  &Converted[1],  &Converted[2],  &Converted[3],
289         &Converted[4],  &Converted[5],  &Converted[6],  &Converted[7],
290         &Converted[8],  &Converted[9],  &Converted[10], &Converted[11],
291         &Converted[12], &Converted[13], &Converted[14], &Converted[15]);

293     /* Pack converted data into a byte array */

295     for (i = 0; i < BytesConverted; i++)
296     {
297         OutputData[i] = (unsigned char) Converted[i];
298     }

300     return ((size_t) BytesConverted);
301 }


304 /******************************************************************************
305  *
306  * FUNCTION:    AxGetTableHeader
307  *
308  * PARAMETERS:  InputFile           - Handle for the input acpidump file
309  *              OutputData          - Where the table header is returned
310  *
311  * RETURN:      The actual number of bytes converted
312  *
313  * DESCRIPTION: Extract and convert an ACPI table header
314  *
315  *****************************************************************************/

317 static size_t
318 AxGetTableHeader (
319     FILE                    *InputFile,
320     unsigned char           *OutputData)
321 {
322     size_t                  BytesConverted;
323     size_t                  TotalConverted = 0;
```



```
324     int                     i;


327     /* Get the full 36 byte ACPI table header, requires 3 input text lines */

329     for (i = 0; i < 3; i++)
330     {
331         if (!fgets (HeaderBuffer, AX_LINE_BUFFER_SIZE, InputFile))
332         {
333             return (TotalConverted);
334         }

336         BytesConverted = AxConvertLine (HeaderBuffer, OutputData);
337         TotalConverted += BytesConverted;
338         OutputData += 16;

340         if (BytesConverted != 16)
341         {
342             return (TotalConverted);
343         }
344     }

346     return (TotalConverted);
347 }


350 /******************************************************************************
351  *
352  * FUNCTION:    AxCountTableInstances
353  *
354  * PARAMETERS:  InputPathname       - Filename for acpidump file
355  *              Signature           - Requested signature to count
356  *
357  * RETURN:      The number of instances of the signature
358  *
359  * DESCRIPTION: Count the instances of tables with the given signature within
360  *              the input acpidump file.
361  *
362  *****************************************************************************/

364 static unsigned int
365 AxCountTableInstances (
366     char                    *InputPathname,
367     char                    *Signature)
368 {
369     FILE                    *InputFile;
370     unsigned int            Instances = 0;


373     InputFile = fopen (InputPathname, "rt");
374     if (!InputFile)
375     {
376         printf ("Could not open file %s\n", InputPathname);
377         return (0);
378     }

380     /* Count the number of instances of this signature */

382     while (fgets (InstanceBuffer, AX_LINE_BUFFER_SIZE, InputFile))
383     {
384         /* Ignore empty lines and lines that start with a space */

386         if (AxIsEmptyLine (InstanceBuffer) ||
387             (InstanceBuffer[0] == ' '))
388         {
389             continue;
```

```
        }

        AxNormalizeSignature (InstanceBuffer);
        if (ACPI_COMPARE_NAME (InstanceBuffer, Signature))
        {
            Instances++;
        }
    }

    fclose (InputFile);
    return (Instances);
}


/******************************************************************************
 *
 * FUNCTION:    AxGetNextInstance
 *
 * PARAMETERS:  InputPathname       - Filename for acpidump file
 *              Signature           - Requested ACPI signature
 *
 * RETURN:      The next instance number for this signature. Zero if this
 *              is the first instance of this signature.
 *
 * DESCRIPTION: Get the next instance number of the specified table. If this
 *              is the first instance of the table, create a new instance
 *              block. Note: only SSDT and PSDT tables can have multiple
 *              instances.
 *
 *****************************************************************************/

static unsigned int
AxGetNextInstance (
    char                    *InputPathname,
    char                    *Signature)
{
    AX_TABLE_INFO           *Info;


    Info = AxTableListHead;
    while (Info)
    {
        if (*(UINT32 *) Signature == Info->Signature)
        {
            break;
        }

        Info = Info->Next;
    }

    if (!Info)
    {
        /* Signature not found, create new table info block */

        Info = malloc (sizeof (AX_TABLE_INFO));
        if (!Info)
        {
            printf ("Could not allocate memory\n");
            exit (0);
        }

        Info->Signature = *(UINT32 *) Signature;
        Info->Instances = AxCountTableInstances (InputPathname, Signature);
        Info->NextInstance = 1;
        Info->Next = AxTableListHead;
        AxTableListHead = Info;
```



```
    }

    if (Info->Instances > 1)
    {
        return (Info->NextInstance++);
    }

    return (0);
}


/******************************************************************************
 *
 * FUNCTION:    AxExtractTables
 *
 * PARAMETERS:  InputPathname       - Filename for acpidump file
 *              Signature           - Requested ACPI signature to extract.
 *                                    NULL means extract ALL tables.
 *              MinimumInstances    - Min instances that are acceptable
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Convert text ACPI tables to binary
 *
 *****************************************************************************/

int
AxExtractTables (
    char                    *InputPathname,
    char                    *Signature,
    unsigned int            MinimumInstances)
{
    FILE                    *InputFile;
    FILE                    *OutputFile = NULL;
    size_t                  BytesWritten;
    size_t                  TotalBytesWritten = 0;
    size_t                  BytesConverted;
    unsigned int            State = AX_STATE_FIND_HEADER;
    unsigned int            FoundTable = 0;
    unsigned int            Instances = 0;
    unsigned int            ThisInstance;
    char                    ThisSignature[4];
    int                     Status = 0;


    /* Open input in text mode, output is in binary mode */

    InputFile = fopen (InputPathname, "rt");
    if (!InputFile)
    {
        printf ("Could not open file %s\n", InputPathname);
        return (-1);
    }

    if (Signature)
    {
        /* Are there enough instances of the table to continue? */

        AxNormalizeSignature (Signature);

        Instances = AxCountTableInstances (InputPathname, Signature);
        if (Instances < MinimumInstances)
        {
            printf ("Table %s was not found in %s\n", Signature, InputPathname);
            Status = -1;
            goto CleanupAndExit;
```

```
522         }

524         if (Instances == 0)
525         {
526             goto CleanupAndExit;
527         }
528     }

530     /* Convert all instances of the table to binary */

532     while (fgets (LineBuffer, AX_LINE_BUFFER_SIZE, InputFile))
533     {
534         switch (State)
535         {
536         case AX_STATE_FIND_HEADER:

538             /* Ignore lines that are too short to be header lines */

540             if (strlen (LineBuffer) < AX_MIN_TABLE_NAME_LENGTH)
541             {
542                 continue;
543             }

545             /* Ignore empty lines and lines that start with a space */

547             if (AxIsEmptyLine (LineBuffer) ||
548                 (LineBuffer[0] == ' '))
549             {
550                 continue;
551             }

553             /*
554              * Ignore lines that are not of the form <sig> @ <addr>.
555              * Examples of lines that must be supported:
556              *
557              * DSDT @ 0x737e4000
558              * XSDT @ 0x737f2fff
559              * RSD PTR @ 0xf6cd0
560              * SSDT @ (nil)
561              */
562             if (!strstr (LineBuffer, " @ "))
563             {
564                 continue;
565             }

567             AxNormalizeSignature (LineBuffer);
568             ACPI_MOVE_NAME (ThisSignature, LineBuffer);

570             if (Signature)
571             {
572                 /* Ignore signatures that don't match */

574                 if (!ACPI_COMPARE_NAME (ThisSignature, Signature))
575                 {
576                     continue;
577                 }
578             }

580             /*
581              * Get the instance number for this signature. Only the
582              * SSDT and PSDT tables can have multiple instances.
583              */
584             ThisInstance = AxGetNextInstance (InputPathname, ThisSignature);

586             /* Build an output filename and create/open the output file */
```



```
588             if (ThisInstance > 0)
589             {
590                 sprintf (Filename, "%4.4s%u.dat", ThisSignature, ThisInstance);
591             }
592             else
593             {
594                 sprintf (Filename, "%4.4s.dat", ThisSignature);
595             }

597             AxStrlwr (Filename);
598             OutputFile = fopen (Filename, "w+b");
599             if (!OutputFile)
600             {
601                 printf ("Could not open file %s\n", Filename);
602                 Status = -1;
603                 goto CleanupAndExit;
604             }

606             State = AX_STATE_EXTRACT_DATA;
607             TotalBytesWritten = 0;
608             FoundTable = 1;
609             continue;

611         case AX_STATE_EXTRACT_DATA:

613             /* Empty line or non-data line terminates the data */

615             if (AxIsEmptyLine (LineBuffer) ||
616                 (LineBuffer[0] != ' '))
617             {
618                 fclose (OutputFile);
619                 OutputFile = NULL;
620                 State = AX_STATE_FIND_HEADER;

622                 printf ("Acpi table [%4.4s] - %u bytes written to %s\n",
623                     ThisSignature, (unsigned int) TotalBytesWritten, Filename);
624                 continue;
625             }

627             /* Convert the ascii data (one line of text) to binary */

629             BytesConverted = AxConvertLine (LineBuffer, Data);

631             /* Write the binary data */

633             BytesWritten = fwrite (Data, 1, BytesConverted, OutputFile);
634             if (BytesWritten != BytesConverted)
635             {
636                 printf ("Error when writing file %s\n", Filename);
637                 fclose (OutputFile);
638                 OutputFile = NULL;
639                 Status = -1;
640                 goto CleanupAndExit;
641             }

643             TotalBytesWritten += BytesConverted;
644             continue;

646         default:

648             Status = -1;
649             goto CleanupAndExit;
650         }
651     }

653     if (!FoundTable)
```

```
654     {
655         printf ("Table %s was not found in %s\n", Signature, InputPathname);
656     }


659 CleanupAndExit:

661     if (OutputFile)
662     {
663         fclose (OutputFile);
664         if (State == AX_STATE_EXTRACT_DATA)
665         {
666             /* Received an EOF while extracting data */

668             printf ("Acpi table [%4.4s] - %u bytes written to %s\n",
669                 ThisSignature, (unsigned int) TotalBytesWritten, Filename);
670         }
671     }

673     fclose (InputFile);
674     return (Status);
675 }


678 /******************************************************************************
679  *
680  * FUNCTION:    AxListTables
681  *
682  * PARAMETERS:  InputPathname       - Filename for acpidump file
683  *
684  * RETURN:      Status
685  *
686  * DESCRIPTION: Display info for all ACPI tables found in input. Does not
687  *              perform an actual extraction of the tables.
688  *
689  *****************************************************************************/

691 int
692 AxListTables (
693     char                    *InputPathname)
694 {
695     FILE                    *InputFile;
696     size_t                  HeaderSize;
697     unsigned char           Header[48];
698     int                     TableCount = 0;
699     ACPI_TABLE_HEADER       *TableHeader = (ACPI_TABLE_HEADER *) (void *) Header


702     /* Open input in text mode, output is in binary mode */

704     InputFile = fopen (InputPathname, "rt");
705     if (!InputFile)
706     {
707         printf ("Could not open file %s\n", InputPathname);
708         return (-1);
709     }

711     /* Dump the headers for all tables found in the input file */

713     printf ("\nSignature  Length     Revision   OemId    OemTableId"
714             "   OemRevision CompilerId CompilerRevision\n\n");

716     while (fgets (LineBuffer, AX_LINE_BUFFER_SIZE, InputFile))
717     {
718         /* Ignore empty lines and lines that start with a space */
```



```
720         if (AxIsEmptyLine (LineBuffer) ||
721             (LineBuffer[0] == ' '))
722         {
723             continue;
724         }

726         /* Get the 36 byte header and display the fields */

728         HeaderSize = AxGetTableHeader (InputFile, Header);
729         if (HeaderSize < 16)
730         {
731             continue;
732         }

734         /* RSDP has an oddball signature and header */

736         if (!strncmp (TableHeader->Signature, "RSD PTR ", 8))
737         {
738             AxCheckAscii ((char *) &Header[9], 6);
739             printf ("%7.4s                           \"%6.6s\"\n", "RSDP", &Heade
740             TableCount++;
741             continue;
742         }

744         /* Minimum size for table with standard header */

746         if (HeaderSize < sizeof (ACPI_TABLE_HEADER))
747         {
748             continue;
749         }

751         /* Signature and Table length */

753         TableCount++;
754         printf ("%7.4s   0x%8.8X", TableHeader->Signature, TableHeader->Length);

756         /* FACS has only signature and length */

758         if (ACPI_COMPARE_NAME (TableHeader->Signature, "FACS"))
759         {
760             printf ("\n");
761             continue;
762         }

764         /* OEM IDs and Compiler IDs */

766         AxCheckAscii (TableHeader->OemId, 6);
767         AxCheckAscii (TableHeader->OemTableId, 8);
768         AxCheckAscii (TableHeader->AslCompilerId, 4);

770         printf ("     0x%2.2X    \"%6.6s\"  \"%8.8s\"   0x%8.8X    \"%4.4s\"
771             TableHeader->Revision, TableHeader->OemId,
772             TableHeader->OemTableId, TableHeader->OemRevision,
773             TableHeader->AslCompilerId, TableHeader->AslCompilerRevision);
774     }

776     printf ("\nFound %u ACPI tables\n", TableCount);
777     fclose (InputFile);
778     return (0);
779 }
```

```
**********************************************************
    5867 Thu Dec 26 13:50:13 2013
new/usr/src/common/acpica/tools/acpixtract/axmain.c
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: axmain - main module for acpixtract utility
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "acpi.h"
#include "accommon.h"
#include "acapps.h"
#include <stdio.h>


static void
DisplayUsage (
    void);

int
AxExtractTables (
    char                    *InputPathname,
    char                    *Signature,
    unsigned int            MinimumInstances);

int
AxListTables (
```



```
    char                    *InputPathname);


/* Options */

#define AX_EXTRACT_ALL          0
#define AX_LIST_ALL             1
#define AX_EXTRACT_SIGNATURE    2
#define AX_EXTRACT_AML_TABLES   3

static int          AxAction = AX_EXTRACT_AML_TABLES; /* DSDT & SSDTs */

#define AX_OPTIONAL_TABLES      0
#define AX_REQUIRED_TABLE       1

#define AX_UTILITY_NAME             "ACPI Binary Table Extraction Utility"
#define AX_SUPPORTED_OPTIONS        "ahls:v"


/******************************************************************************
 *
 * FUNCTION:    DisplayUsage
 *
 * DESCRIPTION: Usage message
 *
 *****************************************************************************/

static void
DisplayUsage (
    void)
{

    ACPI_USAGE_HEADER ("acpixtract [option] <InputFile>");

    ACPI_OPTION ("-a",                  "Extract all tables, not just DSDT/SSDT");
    ACPI_OPTION ("-l",                  "List table summaries, do not extract");
    ACPI_OPTION ("-s <signature>",      "Extract all tables with <signature>");
    ACPI_OPTION ("-v",                  "Display version information");

    printf ("\nExtract binary ACPI tables from text acpidump output\n");
    printf ("Default invocation extracts the DSDT and all SSDTs\n");
}


/******************************************************************************
 *
 * FUNCTION:    main
 *
 * DESCRIPTION: C main function
 *
 *****************************************************************************/

int
main (
    int                     argc,
    char                    *argv[])
{
    char                    *Filename;
    int                     Status;
    int                     j;


    ACPI_DEBUG_INITIALIZE (); /* For debug version only */
    printf (ACPI_COMMON_SIGNON (AX_UTILITY_NAME));

    if (argc < 2)
```

```
128     {
129         DisplayUsage ();
130         return (0);
131     }
132
133     /* Command line options */
134
135     while ((j = AcpiGetopt (argc, argv, AX_SUPPORTED_OPTIONS)) != EOF) switch (j
136     {
137     case 'a':
138
139         AxAction = AX_EXTRACT_ALL;          /* Extract all tables found */
140         break;
141
142     case 'l':
143
144         AxAction = AX_LIST_ALL;             /* List tables only, do not extract
145         break;
146
147     case 's':
148
149         AxAction = AX_EXTRACT_SIGNATURE;    /* Extract only tables with this sig
150         break;
151
152     case 'v': /* -v: (Version): signon already emitted, just exit */
153
154         return (0);
155
156     case 'h':
157     default:
158
159         DisplayUsage ();
160         return (0);
161     }
162
163     /* Input filename is always required */
164
165     Filename = argv[AcpiGbl_Optind];
166     if (!Filename)
167     {
168         printf ("Missing required input filename\n");
169         return (-1);
170     }
171
172     /* Perform requested action */
173
174     switch (AxAction)
175     {
176     case AX_EXTRACT_ALL:
177
178         Status = AxExtractTables (Filename, NULL, AX_OPTIONAL_TABLES);
179         break;
180
181     case AX_LIST_ALL:
182
183         Status = AxListTables (Filename);
184         break;
185
186     case AX_EXTRACT_SIGNATURE:
187
188         Status = AxExtractTables (Filename, AcpiGbl_Optarg, AX_REQUIRED_TABLE);
189         break;
190
191     default:
192         /*
193          * Default output is the DSDT and all SSDTs. One DSDT is required,
```



```
194          * any SSDTs are optional.
195          */
196         Status = AxExtractTables (Filename, "DSDT", AX_REQUIRED_TABLE);
197         if (Status)
198         {
199             return (Status);
200         }
201
202         Status = AxExtractTables (Filename, "SSDT", AX_OPTIONAL_TABLES);
203         break;
204     }
205
206     return (Status);
207 }
```

```
**********************************************************
   13245 Thu Dec 26 13:50:13 2013
new/usr/src/common/acpica/tools/examples/examples.c
update to acpica-unix2-20131218
acpica-unix2-20130823
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: examples - Example ACPICA initialization and execution code
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXAMPLES_C__
#include "examples.h"

#define _COMPONENT          ACPI_EXAMPLE
        ACPI_MODULE_NAME    ("examples")


/******************************************************************************
 *
 * ACPICA Example Code
 *
 * This module contains examples of how the host OS should interface to the
 * ACPICA subsystem.
 *
 * 1) How to use the platform/acenv.h file and how to set configuration
 *      options.
 *
```



```
 * 2) main - using the debug output mechanism and the error/warning output
 *      macros.
 *
 * 3) Two examples of the ACPICA initialization sequence. The first is a
 *      initialization with no "early" ACPI table access. The second shows
 *      how to use ACPICA to obtain the tables very early during kernel
 *      initialization, even before dynamic memory is available.
 *
 * 4) How to invoke a control method, including argument setup and how to
 *      access the return value.
 *
 *****************************************************************************/


/* Local Prototypes */

static ACPI_STATUS
InitializeFullAcpica (void);

static ACPI_STATUS
InstallHandlers (void);

static void
NotifyHandler (
    ACPI_HANDLE             Device,
    UINT32                  Value,
    void                    *Context);

static void
ExecuteMAIN (void);

static void
ExecuteOSI (void);

ACPI_STATUS
InitializeAcpiTables (
    void);

ACPI_STATUS
InitializeAcpi (
    void);


/******************************************************************************
 *
 * FUNCTION:    main
 *
 * PARAMETERS:  argc, argv
 *
 * RETURN:      Status
 *
 * DESCRIPTION: Main routine. Shows the use of the various output macros, as
 *              well as the use of the debug layer/level globals.
 *
 *****************************************************************************/

int ACPI_SYSTEM_XFACE
main (
    int                     argc,
    char                    **argv)
{

    ACPI_DEBUG_INITIALIZE (); /* For debug version only */

    printf (ACPI_COMMON_SIGNON ("ACPI Example Code"));
```

```
127     /* Initialize the local ACPI tables (RSDP/RSDT/XSDT/FADT/DSDT/FACS) */

129     ExInitializeAcpiTables ();

131     /* Initialize the ACPICA subsystem */

133     InitializeFullAcpica ();

135     /* Example warning and error output */

137     ACPI_INFO       ((AE_INFO, "Example ACPICA info message"));
138     ACPI_WARNING    ((AE_INFO, "Example ACPICA warning message"));
139     ACPI_ERROR      ((AE_INFO, "Example ACPICA error message"));
140     ACPI_EXCEPTION  ((AE_INFO, AE_AML_OPERAND_TYPE, "Example ACPICA exception m

142     ExecuteOSI ();
143     ExecuteMAIN ();
144     return (0);
145 }


148 /******************************************************************************
149  *
150  * Example ACPICA initialization code. This shows a full initialization with
151  * no early ACPI table access.
152  *
153  *****************************************************************************/

155 static ACPI_STATUS
156 InitializeFullAcpica (void)
157 {
158     ACPI_STATUS             Status;


161     /* Initialize the ACPICA subsystem */

163     Status = AcpiInitializeSubsystem ();
164     if (ACPI_FAILURE (Status))
165     {
166         ACPI_EXCEPTION ((AE_INFO, Status, "While initializing ACPICA"));
167         return (Status);
168     }

170     /* Initialize the ACPICA Table Manager and get all ACPI tables */

172     ACPI_INFO ((AE_INFO, "Loading ACPI tables"));

174     Status = AcpiInitializeTables (NULL, 16, FALSE);
175     if (ACPI_FAILURE (Status))
176     {
177         ACPI_EXCEPTION ((AE_INFO, Status, "While initializing Table Manager"));
178         return (Status);
179     }

181     /* Create the ACPI namespace from ACPI tables */

183     Status = AcpiLoadTables ();
184     if (ACPI_FAILURE (Status))
185     {
186         ACPI_EXCEPTION ((AE_INFO, Status, "While loading ACPI tables"));
187         return (Status);
188     }

190     /* Install local handlers */

192     Status = InstallHandlers ();
```



```
193     if (ACPI_FAILURE (Status))
194     {
195         ACPI_EXCEPTION ((AE_INFO, Status, "While installing handlers"));
196         return (Status);
197     }

199     /* Initialize the ACPI hardware */

201     Status = AcpiEnableSubsystem (ACPI_FULL_INITIALIZATION);
202     if (ACPI_FAILURE (Status))
203     {
204         ACPI_EXCEPTION ((AE_INFO, Status, "While enabling ACPICA"));
205         return (Status);
206     }

208     /* Complete the ACPI namespace object initialization */

210     Status = AcpiInitializeObjects (ACPI_FULL_INITIALIZATION);
211     if (ACPI_FAILURE (Status))
212     {
213         ACPI_EXCEPTION ((AE_INFO, Status, "While initializing ACPICA objects"));
214         return (Status);
215     }

217     return (AE_OK);
218 }


221 /******************************************************************************
222  *
223  * Example ACPICA initialization code with early ACPI table access. This shows
224  * an initialization that requires early access to ACPI tables (before
225  * kernel dynamic memory is available)
226  *
227  *****************************************************************************/

229 /*
230  * The purpose of this static table array is to avoid the use of kernel
231  * dynamic memory which may not be available during early ACPI table
232  * access.
233  */
234 #define ACPI_MAX_INIT_TABLES    16
235 static ACPI_TABLE_DESC      TableArray[ACPI_MAX_INIT_TABLES];


238 /*
239  * This function would be called early in kernel initialization. After this
240  * is called, all ACPI tables are available to the host.
241  */
242 ACPI_STATUS
243 InitializeAcpiTables (
244     void)
245 {
246     ACPI_STATUS             Status;


249     /* Initialize the ACPICA Table Manager and get all ACPI tables */

251     Status = AcpiInitializeTables (TableArray, ACPI_MAX_INIT_TABLES, TRUE);
252     return (Status);
253 }


256 /*
257  * This function would be called after the kernel is initialized and
258  * dynamic/virtual memory is available. It completes the initialization of
```

```
 * the ACPICA subsystem.
 */
ACPI_STATUS
InitializeAcpi (
    void)
{
    ACPI_STATUS             Status;


    /* Initialize the ACPICA subsystem */

    Status = AcpiInitializeSubsystem ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Copy the root table list to dynamic memory */

    Status = AcpiReallocateRootTable ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Create the ACPI namespace from ACPI tables */

    Status = AcpiLoadTables ();
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Install local handlers */

    Status = InstallHandlers ();
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "While installing handlers"));
        return (Status);
    }

    /* Initialize the ACPI hardware */

    Status = AcpiEnableSubsystem (ACPI_FULL_INITIALIZATION);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    /* Complete the ACPI namespace object initialization */

    Status = AcpiInitializeObjects (ACPI_FULL_INITIALIZATION);
    if (ACPI_FAILURE (Status))
    {
        return (Status);
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * Example ACPICA handler and handler installation
 *
```



```
 *****************************************************************************/

static void
NotifyHandler (
    ACPI_HANDLE                 Device,
    UINT32                      Value,
    void                        *Context)
{

    ACPI_INFO ((AE_INFO, "Received a notify 0x%X", Value));
}


static ACPI_STATUS
InstallHandlers (void)
{
    ACPI_STATUS             Status;


    /* Install global notify handler */

    Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_SYSTEM_NOTIFY,
                                       NotifyHandler, NULL);
    if (ACPI_FAILURE (Status))
    {
        ACPI_EXCEPTION ((AE_INFO, Status, "While installing Notify handler"));
        return (Status);
    }

    return (AE_OK);
}


/******************************************************************************
 *
 * Examples of control method execution.
 *
 * _OSI is a predefined method that is implemented internally within ACPICA.
 *
 * Shows the following elements:
 *
 * 1) How to setup a control method argument and argument list
 * 2) How to setup the return value object
 * 3) How to invoke AcpiEvaluateObject
 * 4) How to check the returned ACPI_STATUS
 * 5) How to analyze the return value
 *
 *****************************************************************************/

static void
ExecuteOSI (void)
{
    ACPI_STATUS             Status;
    ACPI_OBJECT_LIST        ArgList;
    ACPI_OBJECT             Arg[1];
    ACPI_BUFFER             ReturnValue;
    ACPI_OBJECT             *Object;


    ACPI_INFO ((AE_INFO, "Executing _OSI reserved method"));

    /* Setup input argument */

    ArgList.Count = 1;
    ArgList.Pointer = Arg;
```

```
391     Arg[0].Type = ACPI_TYPE_STRING;
392     Arg[0].String.Pointer = "Windows 2001";
393     Arg[0].String.Length = strlen (Arg[0].String.Pointer);

395     /* Ask ACPICA to allocate space for the return object */

397     ReturnValue.Length = ACPI_ALLOCATE_BUFFER;

399     Status = AcpiEvaluateObject (NULL, "\\_OSI", &ArgList, &ReturnValue);
400     if (ACPI_FAILURE (Status))
401     {
402         ACPI_EXCEPTION ((AE_INFO, Status, "While executing _OSI"));
403         return;
404     }

406     /* Ensure that the return object is large enough */

408     if (ReturnValue.Length < sizeof (ACPI_OBJECT))
409     {
410         AcpiOsPrintf ("Return value from _OSI method too small, %.8X\n",
411             ReturnValue.Length);
412         goto ErrorExit;
413     }

415     /* Expect an integer return value from execution of _OSI */

417     Object = ReturnValue.Pointer;
418     if (Object->Type != ACPI_TYPE_INTEGER)
419     {
420         AcpiOsPrintf ("Invalid return type from _OSI, %.2X\n", Object->Type);
421     }

423     ACPI_INFO ((AE_INFO, "_OSI returned 0x%8.8X", (UINT32) Object->Integer.Value


426 ErrorExit:

428     /* Free a buffer created via ACPI_ALLOCATE_BUFFER */

430     AcpiOsFree (ReturnValue.Pointer);
431 }


434 /*****************************************************************************
435  *
436  * Execute an actual control method in the DSDT (MAIN)
437  *
438  ****************************************************************************/

440 static void
441 ExecuteMAIN (void)
442 {
443     ACPI_STATUS             Status;
444     ACPI_OBJECT_LIST        ArgList;
445     ACPI_OBJECT             Arg[1];
446     ACPI_BUFFER             ReturnValue;
447     ACPI_OBJECT             *Object;


450     ACPI_INFO ((AE_INFO, "Executing MAIN method"));

452     /* Setup input argument */

454     ArgList.Count = 1;
455     ArgList.Pointer = Arg;
```



```
457     Arg[0].Type = ACPI_TYPE_STRING;
458     Arg[0].String.Pointer = "Method [MAIN] is executing";
459     Arg[0].String.Length = strlen (Arg[0].String.Pointer);

461     /* Ask ACPICA to allocate space for the return object */

463     ReturnValue.Length = ACPI_ALLOCATE_BUFFER;

465     Status = AcpiEvaluateObject (NULL, "\\MAIN", &ArgList, &ReturnValue);
466     if (ACPI_FAILURE (Status))
467     {
468         ACPI_EXCEPTION ((AE_INFO, Status, "While executing MAIN"));
469         return;
470     }

472     if (ReturnValue.Pointer)
473     {
474         /* Obtain and validate the returned ACPI_OBJECT */

476         Object = ReturnValue.Pointer;
477         if (Object->Type == ACPI_TYPE_STRING)
478         {
479             AcpiOsPrintf ("Method [MAIN] returned: \"%s\"\n", Object->String.Poi
480         }

482         ACPI_FREE (ReturnValue.Pointer);
483     }
484 }
```

```
**********************************************************
    2318 Thu Dec 26 13:50:14 2013
new/usr/src/common/acpica/tools/examples/examples.h
update to acpica-unix2-20131218
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: examples.h - Common include for Examples program
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#ifndef _EXAMPLES_H
#define _EXAMPLES_H

#include "acpi.h"
#include "accommon.h"
#include "acapps.h"
#include "../acpiexec/aecommon.h"

#include <stdio.h>


void
ExInitializeAcpiTables (
    void);

#endif
```

```
**********************************************************
    7187 Thu Dec 26 13:50:14 2013
new/usr/src/common/acpica/tools/examples/exstubs.c
update to acpica-unix2-20131218
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: exstubs - Stub routines for the Example program
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#include "examples.h"

#include <acutils.h>
#include <acevents.h>
#include <acdispat.h>

#define _COMPONENT          ACPI_EXAMPLE
        ACPI_MODULE_NAME    ("exstubs")


/******************************************************************************
 *
 * DESCRIPTION: Stubs used to facilitate linkage of the example program
 *
 *****************************************************************************/


/* Utilities */
```



```
void
AcpiUtSubsystemShutdown (
    void)
{
}

ACPI_STATUS
AcpiUtExecute_STA (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    UINT32                  *StatusFlags)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_HID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_CID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID_LIST **ReturnCidList)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_UID (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecute_SUB (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    ACPI_PNP_DEVICE_ID      **ReturnId)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtExecutePowerMethods (
    ACPI_NAMESPACE_NODE     *DeviceNode,
    const char              **MethodNames,
    UINT8                   MethodCount,
    UINT8                   *OutValues)
{
    return (AE_NOT_IMPLEMENTED);
}

ACPI_STATUS
AcpiUtEvaluateNumericObject (
    char                    *ObjectName,
    ACPI_NAMESPACE_NODE     *DeviceNode,
    UINT64                  *Value)
{
    return (AE_NOT_IMPLEMENTED);
}
```

```
128 ACPI_STATUS
129 AcpiUtGetResourceEndTag (
130     ACPI_OPERAND_OBJECT     *ObjDesc,
131     UINT8                   **EndTag)
132 {
133     return (AE_OK);
134 }


137 /* Hardware manager */

139 UINT32
140 AcpiHwGetMode (
141     void)
142 {
143     return (0);
144 }

146 ACPI_STATUS
147 AcpiHwReadPort (
148     ACPI_IO_ADDRESS         Address,
149     UINT32                  *Value,
150     UINT32                  Width)
151 {
152     return (AE_OK);
153 }

155 ACPI_STATUS
156 AcpiHwWritePort (
157     ACPI_IO_ADDRESS         Address,
158     UINT32                  Value,
159     UINT32                  Width)
160 {
161     return (AE_OK);
162 }


165 /* Event manager */

167 ACPI_STATUS
168 AcpiInstallNotifyHandler (
169     ACPI_HANDLE             Device,
170     UINT32                  HandlerType,
171     ACPI_NOTIFY_HANDLER     Handler,
172     void                    *Context)
173 {
174     return (AE_OK);
175 }

177 ACPI_STATUS
178 AcpiEvInstallXruptHandlers (
179     void)
180 {
181     return (AE_OK);
182 }

184 ACPI_STATUS
185 AcpiEvInitializeEvents (
186     void)
187 {
188     return (AE_OK);
189 }

191 ACPI_STATUS
192 AcpiEvInstallRegionHandlers (
193     void)
```



```
194 {
195     return (AE_OK);
196 }

198 ACPI_STATUS
199 AcpiEvInitializeOpRegions (
200     void)
201 {
202     return (AE_OK);
203 }

205 ACPI_STATUS
206 AcpiEvInitializeRegion (
207     ACPI_OPERAND_OBJECT     *RegionObj,
208     BOOLEAN                 AcpiNsLocked)
209 {
210     return (AE_OK);
211 }

213 #if (!ACPI_REDUCED_HARDWARE)
214 ACPI_STATUS
215 AcpiEvDeleteGpeBlock (
216     ACPI_GPE_BLOCK_INFO     *GpeBlock)
217 {
218     return (AE_OK);
219 }

221 ACPI_STATUS
222 AcpiEnable (
223     void)
224 {
225     return (AE_OK);
226 }
227 #endif /* !ACPI_REDUCED_HARDWARE */

229 void
230 AcpiEvUpdateGpes (
231     ACPI_OWNER_ID           TableOwnerId)
232 {
233 }

235 ACPI_STATUS
236 AcpiEvAddressSpaceDispatch (
237     ACPI_OPERAND_OBJECT     *RegionObj,
238     ACPI_OPERAND_OBJECT     *FieldObj,
239     UINT32                  Function,
240     UINT32                  RegionOffset,
241     UINT32                  BitWidth,
242     UINT64                  *Value)
243 {
244     return (AE_OK);
245 }

247 ACPI_STATUS
248 AcpiEvAcquireGlobalLock (
249     UINT16                  Timeout)
250 {
251     return (AE_OK);
252 }

254 ACPI_STATUS
255 AcpiEvReleaseGlobalLock (
256     void)
257 {
258     return (AE_OK);
259 }
```

```
ACPI_STATUS
AcpiEvQueueNotifyRequest (
    ACPI_NAMESPACE_NODE     *Node,
    UINT32                  NotifyValue)
{
    return (AE_OK);
}

BOOLEAN
AcpiEvIsNotifyObject (
    ACPI_NAMESPACE_NODE     *Node)
{
    return (TRUE);
}


/* Namespace manager */

ACPI_STATUS
AcpiNsCheckReturnValue (
    ACPI_NAMESPACE_NODE         *Node,
    ACPI_EVALUATE_INFO          *Info,
    UINT32                      UserParamCount,
    ACPI_STATUS                 ReturnStatus,
    ACPI_OPERAND_OBJECT         **ReturnObjectPtr)
{
    return (AE_OK);
}

void
AcpiNsCheckArgumentTypes (
    ACPI_EVALUATE_INFO          *Info)
{
    return;
}

void
AcpiNsCheckArgumentCount (
    char                        *Pathname,
    ACPI_NAMESPACE_NODE         *Node,
    UINT32                      UserParamCount,
    const ACPI_PREDEFINED_INFO  *Predefined)
{
    return;
}

void
AcpiNsCheckAcpiCompliance (
    char                        *Pathname,
    ACPI_NAMESPACE_NODE         *Node,
    const ACPI_PREDEFINED_INFO  *Predefined)
{
    return;
}

const ACPI_PREDEFINED_INFO *
AcpiUtMatchPredefinedMethod (
    char                        *Name)
{
    return (NULL);
}

/* OSL interfaces */

ACPI_THREAD_ID
```



```
AcpiOsGetThreadId (
    void)
{
    return (1);
}

ACPI_STATUS
AcpiOsExecute (
    ACPI_EXECUTE_TYPE       Type,
    ACPI_OSD_EXEC_CALLBACK  Function,
    void                    *Context)
{
    return (AE_SUPPORT);
}
```

```
**********************************************************
   21052 Thu Dec 26 13:50:14 2013
new/usr/src/common/acpica/tools/examples/extables.c
update to acpica-unix2-20131218
**********************************************************
```

```
/******************************************************************************
 *
 * Module Name: extables - ACPI tables for Example program
 *
 *****************************************************************************/

/*
 * Copyright (C) 2000 - 2013, Intel Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions, and the following disclaimer,
 *    without modification.
 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
 *    substantially similar to the "NO WARRANTY" disclaimer below
 *    ("Disclaimer") and any redistribution must be conditioned upon
 *    including a substantially similar Disclaimer requirement for further
 *    binary redistribution.
 * 3. Neither the names of the above-listed copyright holders nor the names
 *    of any contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * Alternatively, this software may be distributed under the terms of the
 * GNU General Public License ("GPL") version 2 as published by the Free
 * Software Foundation.
 *
 * NO WARRANTY
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 */

#define __EXTABLES_C__

#include "examples.h"
#include "actables.h"

#define _COMPONENT          ACPI_EXAMPLE
        ACPI_MODULE_NAME    ("extables")

ACPI_PHYSICAL_ADDRESS
AeLocalGetRootPointer (
    void);


/******************************************************************************
 *
 * ACPICA Example tables and table setup
 *
 * This module contains the ACPI tables used for the example program. The
```



```
 * original source code for the tables appears at the end of the module.
 *
 *****************************************************************************/


/* These tables will be modified at runtime */

unsigned char RsdpCode[] =
{
    0x52,0x53,0x44,0x20,0x50,0x54,0x52,0x20,  /* 00000000    "RSD PTR " */
    0x43,0x49,0x4E,0x54,0x45,0x4C,0x20,0x02,  /* 00000008    "CINTEL ." */
    0x00,0x00,0x00,0x00,0x24,0x00,0x00,0x00,  /* 00000010    "....$..." */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000018    "........" */
    0xDC,0x00,0x00,0x00                       /* 00000020    "...."     */
};

unsigned char RsdtCode[] =
{
    0x52,0x53,0x44,0x54,0x28,0x00,0x00,0x00,  /* 00000000    "RSDT(..." */
    0x01,0x10,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00   /* 00000020    "... ...." */
};

unsigned char XsdtCode[] =
{
    0x58,0x53,0x44,0x54,0x2C,0x00,0x00,0x00,  /* 00000000    "XSDT,..." */
    0x01,0x06,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    "..INTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x00,0x00,0x00,0x00                       /* 00000028    "....."    */
};

unsigned char FadtCode[] =
{
    0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00,  /* 00000000    "FACP...." */
    0x05,0x64,0x49,0x4E,0x54,0x45,0x4C,0x20,  /* 00000008    ".dINTEL " */
    0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45,  /* 00000010    "TEMPLATE" */
    0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
    0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00,  /* 00000020    "... ...." */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000038    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000040    "........" */
    0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000048    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000050    "........" */
    0x04,0x02,0x01,0x04,0x08,0x00,0x00,0x00,  /* 00000058    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000060    "........" */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000068    "........" */
    0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01,  /* 00000070    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000078    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000080    "........" */
    0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,  /* 00000088    "........" */
    0x00,0x00,0x00,0x00,0x01,0x20,0x00,0x02,  /* 00000090    "..... .." */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000098    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000A0    "........" */
    0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x02,  /* 000000A8    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B0    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000B8    "........" */
    0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x00,  /* 000000C0    "........" */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000C8    "........" */
    0x01,0x20,0x00,0x03,0x01,0x00,0x00,0x00,  /* 000000D0    ". ......" */
    0x00,0x00,0x00,0x00,0x01,0x40,0x00,0x01,  /* 000000D8    ".....@.." */
    0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E0    "........" */
```

```
128     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000E8    "........" */
129     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01,  /* 000000F0    "........" */
130     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 000000F8    "........" */
131     0x01,0x08,0x00,0x01,0x00,0x00,0x00,0x00,  /* 00000100    "........" */
132     0x00,0x00,0x00,0x00                       /* 00000108    "...."     */
133 };

135 /* Fixed tables */

137 static unsigned char FacsCode[] =
138 {
139     0x46,0x41,0x43,0x53,0x40,0x00,0x00,0x00,  /* 00000000    "FACS@..." */
140     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000008    "........" */
141     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000010    "........" */
142     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000018    "........" */
143     0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000020    "........" */
144     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000028    "........" */
145     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  /* 00000030    "........" */
146     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   /* 00000038    "........" */
147 };

149 static unsigned char DsdtCode[] =
150 {
151     0x44,0x53,0x44,0x54,0x67,0x00,0x00,0x00,  /* 00000000    "DSDTg..." */
152     0x02,0x97,0x49,0x6E,0x74,0x65,0x6C,0x00,  /* 00000008    "..Intel." */
153     0x54,0x65,0x6D,0x70,0x6C,0x61,0x74,0x65,  /* 00000010    "Template" */
154     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C,  /* 00000018    "....INTL" */
155     0x15,0x11,0x13,0x20,0x14,0x42,0x04,0x4D,  /* 00000020    "... .B.M" */
156     0x41,0x49,0x4E,0x01,0x70,0x73,0x0D,0x4D,  /* 00000028    "AIN.ps.M" */
157     0x61,0x69,0x6E,0x2F,0x41,0x72,0x67,0x30,  /* 00000030    "ain/Arg0" */
158     0x3A,0x20,0x00,0x68,0x00,0x5B,0x31,0xA4,  /* 00000038    ": .h.[1." */
159     0x0D,0x4D,0x61,0x69,0x6E,0x20,0x73,0x75,  /* 00000040    ".Main su" */
160     0x63,0x63,0x65,0x73,0x73,0x66,0x75,0x6C,  /* 00000048    "ccessful" */
161     0x6C,0x79,0x20,0x63,0x6F,0x6D,0x70,0x6C,  /* 00000050    "ly compl" */
162     0x65,0x74,0x65,0x64,0x20,0x65,0x78,0x65,  /* 00000058    "eted exe" */
163     0x63,0x75,0x74,0x69,0x6F,0x6E,0x00        /* 00000060    "cution."  */
164 };


167 /* Useful pointers */

169 ACPI_TABLE_RSDP *Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, RsdpCode);
170 ACPI_TABLE_RSDT *Rsdt = ACPI_CAST_PTR (ACPI_TABLE_RSDT, RsdtCode);
171 ACPI_TABLE_XSDT *Xsdt = ACPI_CAST_PTR (ACPI_TABLE_XSDT, XsdtCode);
172 ACPI_TABLE_FADT *Fadt = ACPI_CAST_PTR (ACPI_TABLE_FADT, FadtCode);


175 /******************************************************************************
176  *
177  * Build the various required ACPI tables:
178  *
179  * 1) Setup RSDP to point to the RSDT and XSDT
180  * 2) Setup RSDT/XSDT to point to the FADT
181  * 3) Setup FADT to point to the DSDT and FACS
182  * 4) Update checksums for all modified tables
183  *
184  *****************************************************************************/

186 void
187 ExInitializeAcpiTables (
188     void)
189 {

191     /* Setup RSDP */

193     Rsdp->RsdtPhysicalAddress = (UINT32) ACPI_TO_INTEGER (RsdtCode);
```



```
194     Rsdp->XsdtPhysicalAddress = (UINT64) ACPI_TO_INTEGER (XsdtCode);

196     /* RSDT and XSDT */

198     Rsdt->TableOffsetEntry[0] = (UINT32) ACPI_TO_INTEGER (FadtCode);
199     Xsdt->TableOffsetEntry[0] = (UINT64) ACPI_TO_INTEGER (FadtCode);

201     /* FADT */

203     Fadt->Facs = 0;
204     Fadt->Dsdt = 0;
205     Fadt->XFacs = (UINT64) ACPI_TO_INTEGER (FacsCode);
206     Fadt->XDsdt = (UINT64) ACPI_TO_INTEGER (DsdtCode);

208     /* Set new checksums for the modified tables */

210     Rsdp->Checksum = 0;
211     Rsdp->Checksum = (UINT8) -AcpiTbChecksum (
212         (void *) RsdpCode, ACPI_RSDP_CHECKSUM_LENGTH);

214     Rsdt->Header.Checksum = 0;
215     Rsdt->Header.Checksum = (UINT8) -AcpiTbChecksum (
216         (void *) Rsdt, Rsdt->Header.Length);

218     Xsdt->Header.Checksum = 0;
219     Xsdt->Header.Checksum =  (UINT8) -AcpiTbChecksum (
220         (void *) Xsdt, Xsdt->Header.Length);

222     Fadt->Header.Checksum = 0;
223     Fadt->Header.Checksum =  (UINT8) -AcpiTbChecksum (
224         (void *) Fadt, Fadt->Header.Length);
225 }


228 /******************************************************************************
229  *
230  * OSL support - return the address of the RSDP
231  *
232  *****************************************************************************/

234 ACPI_PHYSICAL_ADDRESS
235 AeLocalGetRootPointer (
236     void)
237 {

239     return ((ACPI_PHYSICAL_ADDRESS) RsdpCode);
240 }


243 #ifdef DO_NOT_COMPILE_ACPI_TABLE_CODE
244 /******************************************************************************
245  *
246  * ACPICA Example table source code
247  *
248  * This is the original source code for the tables above
249  *
250  *****************************************************************************/

252 /* RSDP */

254 [0008]                          Signature : "RSD PTR "
255 [0001]                           Checksum : 43
256 [0006]                             Oem ID : "INTEL "
257 [0001]                           Revision : 02
258 [0004]                       RSDT Address : 00000000
259 [0004]                             Length : 00000024
```

```
[0008]                      XSDT Address : 0000000000000000
[0001]                  Extended Checksum : DC
[0003]                          Reserved : 000000


/* RSDT */

[0004]                         Signature : "RSDT"    [Root System Description T
[0004]                      Table Length : 00000044
[0001]                          Revision : 01
[0001]                          Checksum : B1
[0006]                            Oem ID : "INTEL "
[0008]                      Oem Table ID : "TEMPLATE"
[0004]                      Oem Revision : 00000001
[0004]                   Asl Compiler ID : "INTL"
[0004]             Asl Compiler Revision : 20100528

[0004]            ACPI Table Address   0 : 00000001


/* XSDT */

[0004]                         Signature : "XSDT"    [Extended System Descripti
[0004]                      Table Length : 00000064
[0001]                          Revision : 01
[0001]                          Checksum : 8B
[0006]                            Oem ID : "INTEL "
[0008]                      Oem Table ID : "TEMPLATE"
[0004]                      Oem Revision : 00000001
[0004]                   Asl Compiler ID : "INTL"
[0004]             Asl Compiler Revision : 20100528

[0008]            ACPI Table Address   0 : 0000000000000001


/* FADT */

[0004]                         Signature : "FACP"    [Fixed ACPI Description Ta
[0004]                      Table Length : 0000010C
[0001]                          Revision : 05
[0001]                          Checksum : 18
[0006]                            Oem ID : "INTEL "
[0008]                      Oem Table ID : "TEMPLATE"
[0004]                      Oem Revision : 00000000
[0004]                   Asl Compiler ID : "INTL"
[0004]             Asl Compiler Revision : 20111123

[0004]                      FACS Address : 00000001
[0004]                      DSDT Address : 00000001
[0001]                             Model : 00
[0001]                      PM Profile : 00 [Unspecified]
[0002]                     SCI Interrupt : 0000
[0004]                  SMI Command Port : 00000000
[0001]                 ACPI Enable Value : 00
[0001]                ACPI Disable Value : 00
[0001]                    S4BIOS Command : 00
[0001]                    P-State Control : 00
[0004]            PM1A Event Block Address : 00000001
[0004]            PM1B Event Block Address : 00000000
[0004]           PM1A Control Block Address : 00000001
[0004]           PM1B Control Block Address : 00000000
[0004]            PM2 Control Block Address : 00000001
[0004]               PM Timer Block Address : 00000001
[0004]                   GPE0 Block Address : 00000001
[0004]                   GPE1 Block Address : 00000000
[0001]             PM1 Event Block Length : 04
```



```
[0001]             PM1 Control Block Length : 02
[0001]             PM2 Control Block Length : 01
[0001]                PM Timer Block Length : 04
[0001]                    GPE0 Block Length : 08
[0001]                    GPE1 Block Length : 00
[0001]                     GPE1 Base Offset : 00
[0001]                         _CST Support : 00
[0002]                           C2 Latency : 0000
[0002]                           C3 Latency : 0000
[0002]                      CPU Cache Size : 0000
[0002]                  Cache Flush Stride : 0000
[0001]                     Duty Cycle Offset : 00
[0001]                      Duty Cycle Width : 00
[0001]                  RTC Day Alarm Index : 00
[0001]                RTC Month Alarm Index : 00
[0001]                    RTC Century Index : 00
[0002]           Boot Flags (decoded below) : 0000
                Legacy Devices Supported (V2) : 0
             8042 Present on ports 60/64 (V2) : 0
                       VGA Not Present (V4) : 0
                     MSI Not Supported (V4) : 0
                PCIe ASPM Not Supported (V4) : 0
                   CMOS RTC Not Present (V5) : 0
[0001]                               Reserved : 00
[0004]                    Flags (decoded below) : 00000000
     WBINVD instruction is operational (V1) : 0
              WBINVD flushes all caches (V1) : 0
                    All CPUs support C1 (V1) : 0
                  C2 works on MP system (V1) : 0
             Control Method Power Button (V1) : 0
             Control Method Sleep Button (V1) : 0
         RTC wake not in fixed reg space (V1) : 0
             RTC can wake system from S4 (V1) : 0
                         32-bit PM Timer (V1) : 0
                       Docking Supported (V1) : 0
                Reset Register Supported (V2) : 0
                            Sealed Case (V3) : 0
                    Headless - No Video (V3) : 0
         Use native instr after SLP_TYPx (V3) : 0
               PCIEXP_WAK Bits Supported (V4) : 0
                      Use Platform Timer (V4) : 0
                RTC_STS valid on S4 wake (V4) : 0
                 Remote Power-on capable (V4) : 0
                  Use APIC Cluster Model (V4) : 0
      Use APIC Physical Destination Mode (V4) : 0
                           Hardware Reduced (V5) : 0
                         Low Power S0 Idle (V5) : 0

[0012]                          Reset Register : [Generic Address Structure]
[0001]                               Space ID : 01 [SystemIO]
[0001]                              Bit Width : 08
[0001]                             Bit Offset : 00
[0001]                   Encoded Access Width : 01 [Byte Access:8]
[0008]                               Address : 0000000000000001

[0001]                   Value to cause reset : 00
[0003]                               Reserved : 000000
[0008]                          FACS Address : 0000000000000001
[0008]                          DSDT Address : 0000000000000001
[0012]                      PM1A Event Block : [Generic Address Structure]
[0001]                               Space ID : 01 [SystemIO]
[0001]                              Bit Width : 20
[0001]                             Bit Offset : 00
[0001]                   Encoded Access Width : 02 [Word Access:16]
[0008]                               Address : 0000000000000001
```

```
392 [0012]                  PM1B Event Block : [Generic Address Structure]
393 [0001]                        Space ID : 01 [SystemIO]
394 [0001]                       Bit Width : 00
395 [0001]                      Bit Offset : 00
396 [0001]            Encoded Access Width : 00 [Undefined/Legacy]
397 [0008]                         Address : 0000000000000000

399 [0012]                PM1A Control Block : [Generic Address Structure]
400 [0001]                        Space ID : 01 [SystemIO]
401 [0001]                       Bit Width : 10
402 [0001]                      Bit Offset : 00
403 [0001]            Encoded Access Width : 02 [Word Access:16]
404 [0008]                         Address : 0000000000000001

406 [0012]                PM1B Control Block : [Generic Address Structure]
407 [0001]                        Space ID : 01 [SystemIO]
408 [0001]                       Bit Width : 00
409 [0001]                      Bit Offset : 00
410 [0001]            Encoded Access Width : 00 [Undefined/Legacy]
411 [0008]                         Address : 0000000000000000

413 [0012]                PM2 Control Block : [Generic Address Structure]
414 [0001]                        Space ID : 01 [SystemIO]
415 [0001]                       Bit Width : 08
416 [0001]                      Bit Offset : 00
417 [0001]            Encoded Access Width : 00 [Undefined/Legacy]
418 [0008]                         Address : 0000000000000001

420 [0012]                     PM Timer Block : [Generic Address Structure]
421 [0001]                        Space ID : 01 [SystemIO]
422 [0001]                       Bit Width : 20
423 [0001]                      Bit Offset : 00
424 [0001]            Encoded Access Width : 03 [DWord Access:32]
425 [0008]                         Address : 0000000000000001

427 [0012]                        GPE0 Block : [Generic Address Structure]
428 [0001]                        Space ID : 01 [SystemIO]
429 [0001]                       Bit Width : 40
430 [0001]                      Bit Offset : 00
431 [0001]            Encoded Access Width : 01 [Byte Access:8]
432 [0008]                         Address : 0000000000000001

434 [0012]                        GPE1 Block : [Generic Address Structure]
435 [0001]                        Space ID : 01 [SystemIO]
436 [0001]                       Bit Width : 00
437 [0001]                      Bit Offset : 00
438 [0001]            Encoded Access Width : 00 [Undefined/Legacy]
439 [0008]                         Address : 0000000000000000


442 [0012]             Sleep Control Register : [Generic Address Structure]
443 [0001]                        Space ID : 01 [SystemIO]
444 [0001]                       Bit Width : 08
445 [0001]                      Bit Offset : 00
446 [0001]            Encoded Access Width : 01 [Byte Access:8]
447 [0008]                         Address : 0000000000000000

449 [0012]              Sleep Status Register : [Generic Address Structure]
450 [0001]                        Space ID : 01 [SystemIO]
451 [0001]                       Bit Width : 08
452 [0001]                      Bit Offset : 00
453 [0001]            Encoded Access Width : 01 [Byte Access:8]
454 [0008]                         Address : 0000000000000000


457 /* FACS */
```



```
459 [0004]                           Signature : "FACS"
460 [0004]                              Length : 00000040
461 [0004]                   Hardware Signature : 00000000
462 [0004]           32 Firmware Waking Vector : 00000000
463 [0004]                         Global Lock : 00000000
464 [0004]                     Flags (decoded below) : 00000000
465                        S4BIOS Support Present : 0
466                   64-bit Wake Supported (V2) : 0
467 [0008]           64 Firmware Waking Vector : 0000000000000000
468 [0001]                             Version : 02
469 [0003]                            Reserved : 000000
470 [0004]           OspmFlags (decoded below) : 00000000
471                64-bit Wake Env Required (V2) : 0


474 /* DSDT - ASL code */

476 DefinitionBlock ("dsdt.aml", "DSDT", 2, "Intel", "Template", 0x00000001)
477 {
478     Method (MAIN, 1, NotSerialized)
479     {
480         Store (Concatenate ("Main/Arg0: ", Arg0), Debug)
481         Return ("Main successfully completed execution")
482     }
483 }
484 #endif
```

```
**********************************************************
    1032 Thu Dec 26 13:50:14 2013
new/usr/src/pkg/manifests/diagnostic-acpitools.mf
PANKOVs restructure
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

set name=pkg.fmri value=pkg:/diagnostic/acpitools@$(PKGVERS)
set name=pkg.description value=acpitools
set name=pkg.summary value=acpitools
set name=info.classification \
    value="org.opensolaris.category.2008:Applications/System Utilities"
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/bin
file path=usr/bin/acpiexec mode=0555
file path=usr/bin/acpixtract mode=0555
file path=usr/bin/iasl mode=0555
license lic_CDDL license=lic_CDDL
license usr/src/common/acpica/THIRDPARTYLICENSE \
    license=usr/src/common/acpica/THIRDPARTYLICENSE
```

```
**********************************************************
   45175 Thu Dec 26 13:50:15 2013
new/usr/src/pkg/manifests/system-kernel.mf
acpica-unix2-20130823
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# The default for payload-bearing actions in this package is to appear in the
# global zone only.  See the include file for greater detail, as well as
# information about overriding the defaults.
#
<include global_zone_only_component>
<include system-kernel.man1m.inc>
<include system-kernel.man2.inc>
<include system-kernel.man4.inc>
<include system-kernel.man5.inc>
<include system-kernel.man7.inc>
<include system-kernel.man7d.inc>
<include system-kernel.man7fs.inc>
<include system-kernel.man7m.inc>
<include system-kernel.man7p.inc>
<include system-kernel.man9.inc>
<include system-kernel.man9e.inc>
<include system-kernel.man9f.inc>
<include system-kernel.man9p.inc>
<include system-kernel.man9s.inc>
set name=pkg.fmri value=pkg:/system/kernel@$(PKGVERS)
set name=pkg.description \
    value="core kernel software for a specific instruction-set architecture"
set name=pkg.summary value="Core Solaris Kernel"
set name=info.classification value=org.opensolaris.category.2008:System/Core
set name=variant.arch value=$(ARCH)
dir path=boot group=sys
$(i386_ONLY)dir path=boot/acpi group=sys
$(i386_ONLY)dir path=boot/acpi/tables group=sys
dir path=boot/solaris group=sys
dir path=boot/solaris/bin group=sys
dir path=etc group=sys
dir path=etc/crypto group=sys
dir path=etc/sock2path.d group=sys
dir path=kernel group=sys
$(i386_ONLY)dir path=kernel/$(ARCH64) group=sys
```



```
dir path=kernel/crypto group=sys
dir path=kernel/crypto/$(ARCH64) group=sys
dir path=kernel/dacf group=sys
dir path=kernel/dacf/$(ARCH64) group=sys
dir path=kernel/drv group=sys
dir path=kernel/drv/$(ARCH64) group=sys
dir path=kernel/exec group=sys
dir path=kernel/exec/$(ARCH64) group=sys
dir path=kernel/fs group=sys
dir path=kernel/fs/$(ARCH64) group=sys
dir path=kernel/ipp group=sys
dir path=kernel/ipp/$(ARCH64) group=sys
dir path=kernel/kiconv group=sys
dir path=kernel/kiconv/$(ARCH64) group=sys
dir path=kernel/mac group=sys
dir path=kernel/mac/$(ARCH64) group=sys
dir path=kernel/misc group=sys
dir path=kernel/misc/$(ARCH64) group=sys
dir path=kernel/misc/scsi_vhci group=sys
dir path=kernel/misc/scsi_vhci/$(ARCH64) group=sys
dir path=kernel/sched group=sys
dir path=kernel/sched/$(ARCH64) group=sys
dir path=kernel/socketmod group=sys
dir path=kernel/socketmod/$(ARCH64) group=sys
dir path=kernel/strmod group=sys
dir path=kernel/strmod/$(ARCH64) group=sys
dir path=kernel/sys group=sys
dir path=kernel/sys/$(ARCH64) group=sys
dir path=lib
dir path=lib/svc
dir path=lib/svc/manifest group=sys
dir path=lib/svc/manifest/system group=sys
dir path=lib/svc/method
dir path=usr/share/man
dir path=usr/share/man/man1m
dir path=usr/share/man/man2
dir path=usr/share/man/man3
dir path=usr/share/man/man4
dir path=usr/share/man/man5
dir path=usr/share/man/man7d
dir path=usr/share/man/man7fs
dir path=usr/share/man/man7m
dir path=usr/share/man/man7p
dir path=usr/share/man/man9
dir path=usr/share/man/man9e
dir path=usr/share/man/man9f
dir path=usr/share/man/man9p
dir path=usr/share/man/man9s
$(i386_ONLY)driver name=acpi_drv perms="* 0666 root sys"
driver name=aggr perms="* 0666 root sys"
driver name=arp perms="arp 0666 root sys"
driver name=bl perms="* 0666 root sys"
driver name=bridge clone_perms="bridge 0666 root sys" \
    policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
$(sparc_ONLY)driver name=bscbus alias=SUNW,bscbus
$(i386_ONLY)driver name=bscbus alias=SVI0101
$(sparc_ONLY)driver name=bscv alias=SUNW,bscv perms="* 0644 root sys"
$(i386_ONLY)driver name=bscv
driver name=clone
driver name=cn perms="* 0620 root tty"
driver name=conskbd perms="kbd 0666 root sys"
driver name=consms perms="mouse 0666 root sys"
driver name=cpuid perms="self 0644 root sys"
$(i386_ONLY)driver name=cpunex alias=cpus
driver name=crypto perms="crypto 0666 root sys"
driver name=cryptoadm perms="cryptoadm 0644 root sys"
```

```
$(sparc_ONLY)driver name=dad alias=ide-disk perms="* 0640 root sys"
driver name=devinfo perms="devinfo 0640 root sys" \
    perms="devinfo,ro 0444 root sys"
driver name=dld perms="* 0666 root sys"
driver name=dlpistub perms="* 0666 root sys"
$(sparc_ONLY)driver name=i8042 alias=8042
$(i386_ONLY)driver name=i8042
driver name=icmp perms="icmp 0666 root sys" \
    policy="read_priv_set=net_icmpaccess write_priv_set=net_icmpaccess"
driver name=icmp6 perms="icmp6 0666 root sys" \
    policy="read_priv_set=net_icmpaccess write_priv_set=net_icmpaccess"
$(i386_ONLY)driver name=intel_nb5000 \
    alias=pci8086,25c0 \
    alias=pci8086,25d0 \
    alias=pci8086,25d4 \
    alias=pci8086,25d8 \
    alias=pci8086,3600 \
    alias=pci8086,4000 \
    alias=pci8086,4001 \
    alias=pci8086,4003 \
    alias=pci8086,65c0
$(i386_ONLY)driver name=intel_nhm \
    alias=pci8086,3423 \
    alias=pci8086,372a
$(i386_ONLY)driver name=intel_nhmex alias=pci8086,3438
driver name=ip perms="ip 0666 root sys" \
    policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
driver name=ip6 perms="ip6 0666 root sys" \
    policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
driver name=ipnet perms="lo0 0666 root sys" \
    policy="read_priv_set=net_observability write_priv_set=net_observability"
driver name=ippctl
driver name=ipsecah perms="ipsecah 0666 root sys" \
    policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
driver name=ipsecesp perms="ipsecesp 0666 root sys" \
    policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
driver name=iptun
driver name=iwscn
driver name=kb8042 alias=pnpPNP,303
driver name=keysock perms="keysock 0666 root sys" \
    policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
driver name=kmdb
driver name=kssl perms="* 0666 root sys"
driver name=llc1 clone_perms="llc1 0666 root sys"
driver name=lofi perms="* 0600 root sys" perms="ctl 0644 root sys"
driver name=log perms="conslog 0666 root sys" perms="log 0640 root sys"
$(i386_ONLY)driver name=mc-amd \
    alias=pci1022,1100 \
    alias=pci1022,1101 \
    alias=pci1022,1102
driver name=mm perms="allkmem 0600 root sys" perms="kmem 0640 root sys" \
    perms="mem 0640 root sys" perms="null 0666 root sys" \
    perms="zero 0666 root sys" \
    policy="allkmem read_priv_set=all write_priv_set=all" \
    policy="kmem read_priv_set=none write_priv_set=all" \
    policy="mem read_priv_set=none write_priv_set=all"
driver name=mouse8042 alias=pnpPNP,f03
$(i386_ONLY)driver name=mpt class=scsi \
    alias=pci1000,30 \
    alias=pci1000,50 \
    alias=pci1000,54 \
    alias=pci1000,56 \
    alias=pci1000,58 \
    alias=pci1000,62 \
    alias=pciex1000,56 \
    alias=pciex1000,58 \
```



```
    alias=pciex1000,62
driver name=nulldriver \
    alias=scsa,nodev \
    alias=scsa,probe
driver name=openeepr perms="openprom 0640 root sys" policy=write_priv_set=all
driver name=options
$(sparc_ONLY)driver name=pci_pci class=pci \
    alias=pci1011,1 \
    alias=pci1011,21 \
    alias=pci1011,24 \
    alias=pci1011,25 \
    alias=pci1011,26 \
    alias=pci1014,22 \
    alias=pciclass,060400
$(i386_ONLY)driver name=pci_pci class=pci \
    alias=pci1011,1 \
    alias=pci1011,21 \
    alias=pci1014,22 \
    alias=pciclass,060400 \
    alias=pciclass,060401
$(sparc_ONLY)driver name=pcieb \
    alias=pciex108e,9010 \
    alias=pciex108e,9020 \
    alias=pciex10b5,8114 \
    alias=pciex10b5,8516 \
    alias=pciex10b5,8517 \
    alias=pciex10b5,8518 \
    alias=pciex10b5,8532 \
    alias=pciex10b5,8533 \
    alias=pciex10b5,8548 \
    alias=pciexclass,060400
$(i386_ONLY)driver name=pcieb \
    alias=pciexclass,060400 \
    alias=pciexclass,060401
$(sparc_ONLY)driver name=pcieb_bcm alias=pciex1166,103
driver name=physmem perms="* 0600 root sys"
driver name=poll perms="* 0666 root sys"
$(sparc_ONLY)driver name=power alias=ali1535d+-power
$(i386_ONLY)driver name=power
driver name=pseudo alias=zconsnex
driver name=ptc perms="* 0666 root sys"
driver name=ptsl perms="* 0666 root sys"
$(sparc_ONLY)driver name=ramdisk alias=SUNW,ramdisk perms="* 0600 root sys" \
    perms="ctl 0644 root sys"
$(i386_ONLY)driver name=ramdisk perms="* 0600 root sys" \
    perms="ctl 0644 root sys"
driver name=random perms="* 0644 root sys" policy=write_priv_set=sys_devices
driver name=rts perms="rts 0666 root sys"
driver name=sad perms="admin 0666 root sys" perms="user 0666 root sys"
driver name=scsi_vhci class=scsi-self-identifying perms="* 0666 root sys" \
    policy="devctl write_priv_set=sys_devices"
$(sparc_ONLY)driver name=sd perms="* 0640 root sys" \
    alias=ide-cdrom \
    alias=scsiclass,00 \
    alias=scsiclass,05
$(i386_ONLY)driver name=sd perms="* 0640 root sys" \
    alias=scsiclass,00 \
    alias=scsiclass,05
driver name=sgen perms="* 0600 root sys" \
    alias=scsa,08.bfcp \
    alias=scsa,08.bvhci
driver name=simnet clone_perms="simnet 0666 root sys" perms="* 0666 root sys"
$(i386_ONLY)driver name=smbios perms="smbios 0444 root sys"
driver name=softmac
driver name=spdsock perms="spdsock 0666 root sys" \
    policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
```

```
260 driver name=st alias=scsiclass,01 perms="* 0666 root sys"
261 driver name=sy perms="tty 0666 root tty"
262 driver name=sysevent perms="* 0600 root sys"
263 driver name=sysmsg perms="msglog 0600 root sys" perms="sysmsg 0600 root sys"
264 driver name=tcp perms="tcp 0666 root sys"
265 driver name=tcp6 perms="tcp6 0666 root sys"
266 driver name=tl perms="* 0666 root sys" clone_perms="ticlts 0666 root sys" \
267     clone_perms="ticots 0666 root sys" clone_perms="ticotsord 0666 root sys"
268 $(sparc_ONLY)driver name=ttymux alias=multiplexer
269 $(i386_ONLY)driver name=tzmon
270 $(sparc_ONLY)driver name=uata \
271     alias=pci1095,646 \
272     alias=pci1095,649 \
273     alias=pci1095,680 \
274     alias=pci10b9,5229 \
275     alias=pci10b9,5288 class=dada class=scsi
276 $(i386_ONLY)driver name=ucode perms="* 0644 root sys"
277 driver name=udp perms="udp 0666 root sys"
278 driver name=udp6 perms="udp6 0666 root sys"
279 $(i386_ONLY)driver name=vgatext \
280     alias=pciclass,000100 \
281     alias=pciclass,030000 \
282     alias=pciclass,030001 \
283     alias=pnpPNP,900
284 driver name=vnic clone_perms="vnic 0666 root sys" perms="* 0666 root sys"
285 driver name=wc perms="* 0600 root sys"
286 $(i386_ONLY)file path=boot/solaris/bin/create_diskmap group=sys mode=0555
287 file path=boot/solaris/bin/create_ramdisk group=sys mode=0555
288 file path=boot/solaris/bin/extract_boot_filelist group=sys mode=0555
289 $(i386_ONLY)file path=boot/solaris/bin/mbr group=sys mode=0555
290 $(i386_ONLY)file path=boot/solaris/bin/symdef group=sys mode=0555
291 $(i386_ONLY)file path=boot/solaris/bin/update_grub group=sys mode=0555
292 file path=boot/solaris/filelist.ramdisk group=sys
293 file path=boot/solaris/filelist.safe group=sys
294 file path=etc/crypto/kcf.conf group=sys \
295     original_name=SUNWckr:etc/crypto/kcf.conf preserve=true
296 $(i386_ONLY)file path=etc/mach group=sys original_name=SUNWos86r:etc/mach \
297     preserve=true
298 file path=etc/name_to_sysnum group=sys \
299     original_name=SUNWckr:etc/name_to_sysnum preserve=renameold
300 file path=etc/sock2path.d/system%2Fkernel group=sys
301 file path=etc/system group=sys original_name=SUNWckr:etc/system preserve=true
302 $(i386_ONLY)file path=kernel/$(ARCH64)/genunix group=sys mode=0755
303 file path=kernel/crypto/$(ARCH64)/aes group=sys mode=0755
304 file path=kernel/crypto/$(ARCH64)/arcfour group=sys mode=0755
305 file path=kernel/crypto/$(ARCH64)/blowfish group=sys mode=0755
306 file path=kernel/crypto/$(ARCH64)/des group=sys mode=0755
307 file path=kernel/crypto/$(ARCH64)/ecc group=sys mode=0755
308 file path=kernel/crypto/$(ARCH64)/md4 group=sys mode=0755
309 file path=kernel/crypto/$(ARCH64)/md5 group=sys mode=0755
310 file path=kernel/crypto/$(ARCH64)/rsa group=sys mode=0755
311 file path=kernel/crypto/$(ARCH64)/sha1 group=sys mode=0755
312 file path=kernel/crypto/$(ARCH64)/sha2 group=sys mode=0755
313 file path=kernel/crypto/$(ARCH64)/swrand group=sys mode=0755
314 $(i386_ONLY)file path=kernel/crypto/aes group=sys mode=0755
315 $(i386_ONLY)file path=kernel/crypto/arcfour group=sys mode=0755
316 $(i386_ONLY)file path=kernel/crypto/blowfish group=sys mode=0755
317 $(i386_ONLY)file path=kernel/crypto/des group=sys mode=0755
318 $(i386_ONLY)file path=kernel/crypto/ecc group=sys mode=0755
319 $(i386_ONLY)file path=kernel/crypto/md4 group=sys mode=0755
320 $(i386_ONLY)file path=kernel/crypto/md5 group=sys mode=0755
321 $(i386_ONLY)file path=kernel/crypto/rsa group=sys mode=0755
322 $(i386_ONLY)file path=kernel/crypto/sha1 group=sys mode=0755
323 $(i386_ONLY)file path=kernel/crypto/sha2 group=sys mode=0755
324 $(i386_ONLY)file path=kernel/crypto/swrand group=sys mode=0755
325 $(sparc_ONLY)file path=kernel/dacf/$(ARCH64)/consconfig_dacf group=sys \
```



```
326     mode=0755
327 file path=kernel/dacf/$(ARCH64)/net_dacf group=sys mode=0755
328 $(i386_ONLY)file path=kernel/dacf/net_dacf group=sys mode=0755
329 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/acpi_drv group=sys
330 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/acpi_toshiba group=sys
331 file path=kernel/drv/$(ARCH64)/aggr group=sys
332 file path=kernel/drv/$(ARCH64)/arp group=sys
333 file path=kernel/drv/$(ARCH64)/bl group=sys
334 file path=kernel/drv/$(ARCH64)/bridge group=sys
335 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/bscbus group=sys
336 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/bscv group=sys
337 file path=kernel/drv/$(ARCH64)/clone group=sys
338 file path=kernel/drv/$(ARCH64)/cn group=sys
339 file path=kernel/drv/$(ARCH64)/conskbd group=sys
340 file path=kernel/drv/$(ARCH64)/consms group=sys
341 file path=kernel/drv/$(ARCH64)/cpuid group=sys
342 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/cpunex group=sys
343 file path=kernel/drv/$(ARCH64)/crypto group=sys
344 file path=kernel/drv/$(ARCH64)/cryptoadm group=sys
345 $(sparc_ONLY)file path=kernel/drv/$(ARCH64)/dad group=sys
346 file path=kernel/drv/$(ARCH64)/devinfo group=sys
347 file path=kernel/drv/$(ARCH64)/dld group=sys
348 file path=kernel/drv/$(ARCH64)/dlpistub group=sys
349 file path=kernel/drv/$(ARCH64)/i8042 group=sys
350 file path=kernel/drv/$(ARCH64)/icmp group=sys
351 file path=kernel/drv/$(ARCH64)/icmp6 group=sys
352 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/intel_nb5000 group=sys
353 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/intel_nhm group=sys
354 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/intel_nhmex group=sys
355 file path=kernel/drv/$(ARCH64)/ip group=sys
356 file path=kernel/drv/$(ARCH64)/ip6 group=sys
357 file path=kernel/drv/$(ARCH64)/ipnet group=sys
358 file path=kernel/drv/$(ARCH64)/ippctl group=sys
359 file path=kernel/drv/$(ARCH64)/ipsecah group=sys
360 file path=kernel/drv/$(ARCH64)/ipsecesp group=sys
361 file path=kernel/drv/$(ARCH64)/iptun group=sys
362 file path=kernel/drv/$(ARCH64)/iwscn group=sys
363 file path=kernel/drv/$(ARCH64)/kb8042 group=sys
364 file path=kernel/drv/$(ARCH64)/keysock group=sys
365 file path=kernel/drv/$(ARCH64)/kmdb group=sys
366 file path=kernel/drv/$(ARCH64)/kssl group=sys
367 file path=kernel/drv/$(ARCH64)/llc1 group=sys
368 file path=kernel/drv/$(ARCH64)/lofi group=sys
369 file path=kernel/drv/$(ARCH64)/log group=sys
370 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/mc-amd group=sys
371 file path=kernel/drv/$(ARCH64)/mm group=sys
372 file path=kernel/drv/$(ARCH64)/mouse8042 group=sys
373 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/mpt group=sys
374 file path=kernel/drv/$(ARCH64)/nulldriver group=sys
375 file path=kernel/drv/$(ARCH64)/openeepr group=sys
376 file path=kernel/drv/$(ARCH64)/options group=sys
377 file path=kernel/drv/$(ARCH64)/pci_pci group=sys
378 file path=kernel/drv/$(ARCH64)/pcieb group=sys
379 $(sparc_ONLY)file path=kernel/drv/$(ARCH64)/pcieb_bcm group=sys
380 file path=kernel/drv/$(ARCH64)/physmem group=sys
381 file path=kernel/drv/$(ARCH64)/poll group=sys
382 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/power group=sys
383 file path=kernel/drv/$(ARCH64)/pseudo group=sys
384 file path=kernel/drv/$(ARCH64)/ptc group=sys
385 file path=kernel/drv/$(ARCH64)/ptsl group=sys
386 file path=kernel/drv/$(ARCH64)/ramdisk group=sys
387 file path=kernel/drv/$(ARCH64)/random group=sys
388 file path=kernel/drv/$(ARCH64)/rts group=sys
389 file path=kernel/drv/$(ARCH64)/sad group=sys
390 file path=kernel/drv/$(ARCH64)/scsi_vhci group=sys
391 file path=kernel/drv/$(ARCH64)/sd group=sys
```

```
392 file path=kernel/drv/$(ARCH64)/sgen group=sys
393 file path=kernel/drv/$(ARCH64)/simnet group=sys
394 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/smbios group=sys
395 file path=kernel/drv/$(ARCH64)/softmac group=sys
396 file path=kernel/drv/$(ARCH64)/spdsock group=sys
397 file path=kernel/drv/$(ARCH64)/st group=sys
398 file path=kernel/drv/$(ARCH64)/sy group=sys
399 file path=kernel/drv/$(ARCH64)/sysevent group=sys
400 file path=kernel/drv/$(ARCH64)/sysmsg group=sys
401 file path=kernel/drv/$(ARCH64)/tcp group=sys
402 file path=kernel/drv/$(ARCH64)/tcp6 group=sys
403 file path=kernel/drv/$(ARCH64)/tl group=sys
404 $(sparc_ONLY)file path=kernel/drv/$(ARCH64)/ttymux group=sys
405 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/tzmon group=sys
406 $(sparc_ONLY)file path=kernel/drv/$(ARCH64)/uata group=sys
407 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/ucode group=sys
408 file path=kernel/drv/$(ARCH64)/udp group=sys
409 file path=kernel/drv/$(ARCH64)/udp6 group=sys
410 $(i386_ONLY)file path=kernel/drv/$(ARCH64)/vgatext group=sys
411 file path=kernel/drv/$(ARCH64)/vnic group=sys
412 file path=kernel/drv/$(ARCH64)/wc group=sys
413 $(i386_ONLY)file path=kernel/drv/acpi_drv group=sys
414 $(i386_ONLY)file path=kernel/drv/acpi_drv.conf group=sys
415 $(i386_ONLY)file path=kernel/drv/acpi_toshiba group=sys
416 $(i386_ONLY)file path=kernel/drv/aggr group=sys
417 file path=kernel/drv/aggr.conf group=sys
418 $(i386_ONLY)file path=kernel/drv/arp group=sys
419 file path=kernel/drv/arp.conf group=sys
420 $(i386_ONLY)file path=kernel/drv/bl group=sys
421 file path=kernel/drv/bl.conf group=sys
422 $(i386_ONLY)file path=kernel/drv/bridge group=sys
423 file path=kernel/drv/bridge.conf group=sys
424 $(i386_ONLY)file path=kernel/drv/bscbus group=sys
425 $(i386_ONLY)file path=kernel/drv/bscbus.conf group=sys
426 $(i386_ONLY)file path=kernel/drv/bscv group=sys
427 $(i386_ONLY)file path=kernel/drv/bscv.conf group=sys
428 $(i386_ONLY)file path=kernel/drv/clone group=sys
429 file path=kernel/drv/clone.conf group=sys
430 $(i386_ONLY)file path=kernel/drv/cn group=sys
431 file path=kernel/drv/cn.conf group=sys
432 $(i386_ONLY)file path=kernel/drv/conskbd group=sys
433 file path=kernel/drv/conskbd.conf group=sys
434 $(i386_ONLY)file path=kernel/drv/consms group=sys
435 file path=kernel/drv/consms.conf group=sys
436 $(i386_ONLY)file path=kernel/drv/cpuid group=sys
437 file path=kernel/drv/cpuid.conf group=sys
438 $(i386_ONLY)file path=kernel/drv/cpunex group=sys
439 $(i386_ONLY)file path=kernel/drv/crypto group=sys
440 file path=kernel/drv/crypto.conf group=sys
441 $(i386_ONLY)file path=kernel/drv/cryptoadm group=sys
442 file path=kernel/drv/cryptoadm.conf group=sys
443 $(sparc_ONLY)file path=kernel/drv/dad.conf group=sys
444 $(i386_ONLY)file path=kernel/drv/devinfo group=sys
445 file path=kernel/drv/devinfo.conf group=sys
446 $(i386_ONLY)file path=kernel/drv/dld group=sys
447 file path=kernel/drv/dld.conf group=sys
448 $(i386_ONLY)file path=kernel/drv/dlpistub group=sys
449 file path=kernel/drv/dlpistub.conf group=sys
450 $(i386_ONLY)file path=kernel/drv/i8042 group=sys
451 $(i386_ONLY)file path=kernel/drv/icmp group=sys
452 file path=kernel/drv/icmp.conf group=sys
453 $(i386_ONLY)file path=kernel/drv/icmp6 group=sys
454 file path=kernel/drv/icmp6.conf group=sys
455 $(i386_ONLY)file path=kernel/drv/intel_nb5000 group=sys
456 $(i386_ONLY)file path=kernel/drv/intel_nb5000.conf group=sys
457 $(i386_ONLY)file path=kernel/drv/intel_nhm group=sys
```



```
458 $(i386_ONLY)file path=kernel/drv/intel_nhm.conf group=sys
459 $(i386_ONLY)file path=kernel/drv/intel_nhmex group=sys
460 $(i386_ONLY)file path=kernel/drv/intel_nhmex.conf group=sys
461 $(i386_ONLY)file path=kernel/drv/ip group=sys
462 file path=kernel/drv/ip.conf group=sys
463 $(i386_ONLY)file path=kernel/drv/ip6 group=sys
464 file path=kernel/drv/ip6.conf group=sys
465 $(i386_ONLY)file path=kernel/drv/ipnet group=sys
466 file path=kernel/drv/ipnet.conf group=sys
467 $(i386_ONLY)file path=kernel/drv/ippctl group=sys
468 file path=kernel/drv/ippctl.conf group=sys
469 $(i386_ONLY)file path=kernel/drv/ipsecah group=sys
470 file path=kernel/drv/ipsecah.conf group=sys
471 $(i386_ONLY)file path=kernel/drv/ipsecesp group=sys
472 file path=kernel/drv/ipsecesp.conf group=sys
473 $(i386_ONLY)file path=kernel/drv/iptun group=sys
474 file path=kernel/drv/iptun.conf group=sys
475 $(i386_ONLY)file path=kernel/drv/iwscn group=sys
476 file path=kernel/drv/iwscn.conf group=sys
477 $(i386_ONLY)file path=kernel/drv/kb8042 group=sys
478 $(i386_ONLY)file path=kernel/drv/keysock group=sys
479 file path=kernel/drv/keysock.conf group=sys
480 $(i386_ONLY)file path=kernel/drv/kmdb group=sys
481 file path=kernel/drv/kmdb.conf group=sys
482 $(i386_ONLY)file path=kernel/drv/kssl group=sys
483 file path=kernel/drv/kssl.conf group=sys
484 $(i386_ONLY)file path=kernel/drv/llc1 group=sys
485 file path=kernel/drv/llc1.conf group=sys
486 $(i386_ONLY)file path=kernel/drv/lofi group=sys
487 file path=kernel/drv/lofi.conf group=sys
488 $(i386_ONLY)file path=kernel/drv/log group=sys
489 file path=kernel/drv/log.conf group=sys \
490     original_name=SUNWckr:kernel/drv/log.conf preserve=true
491 $(i386_ONLY)file path=kernel/drv/mc-amd group=sys
492 $(i386_ONLY)file path=kernel/drv/mc-amd.conf group=sys
493 $(i386_ONLY)file path=kernel/drv/mm group=sys
494 file path=kernel/drv/mm.conf group=sys
495 $(i386_ONLY)file path=kernel/drv/mouse8042 group=sys
496 $(i386_ONLY)file path=kernel/drv/mpt group=sys
497 $(i386_ONLY)file path=kernel/drv/mpt.conf group=sys \
498     original_name=SUNWckr:kernel/drv/mpt.conf preserve=true
499 $(i386_ONLY)file path=kernel/drv/nulldriver group=sys
500 $(i386_ONLY)file path=kernel/drv/openeepr group=sys
501 file path=kernel/drv/openeepr.conf group=sys
502 $(i386_ONLY)file path=kernel/drv/options group=sys
503 file path=kernel/drv/options.conf group=sys
504 $(i386_ONLY)file path=kernel/drv/pci_pci group=sys
505 $(i386_ONLY)file path=kernel/drv/pcieb group=sys
506 file path=kernel/drv/pcieb.conf group=sys
507 $(i386_ONLY)file path=kernel/drv/physmem group=sys
508 file path=kernel/drv/physmem.conf group=sys
509 $(i386_ONLY)file path=kernel/drv/poll group=sys
510 file path=kernel/drv/poll.conf group=sys
511 $(i386_ONLY)file path=kernel/drv/power group=sys
512 $(i386_ONLY)file path=kernel/drv/power.conf group=sys
513 $(i386_ONLY)file path=kernel/drv/pseudo group=sys
514 file path=kernel/drv/pseudo.conf group=sys
515 $(i386_ONLY)file path=kernel/drv/ptc group=sys
516 file path=kernel/drv/ptc.conf group=sys
517 $(i386_ONLY)file path=kernel/drv/ptsl group=sys
518 file path=kernel/drv/ptsl.conf group=sys
519 $(i386_ONLY)file path=kernel/drv/ramdisk group=sys
520 file path=kernel/drv/ramdisk.conf group=sys
521 $(i386_ONLY)file path=kernel/drv/random group=sys
522 file path=kernel/drv/random.conf group=sys
523 $(i386_ONLY)file path=kernel/drv/rts group=sys
```

```
524 file path=kernel/drv/rts.conf group=sys
525 $(i386_ONLY)file path=kernel/drv/sad group=sys
526 file path=kernel/drv/sad.conf group=sys
527 $(i386_ONLY)file path=kernel/drv/scsi_vhci group=sys
528 file path=kernel/drv/scsi_vhci.conf group=sys \
529     original_name=SUNWckr:kernel/drv/scsi_vhci.conf preserve=true
530 $(i386_ONLY)file path=kernel/drv/sd group=sys
531 $(sparc_ONLY)file path=kernel/drv/sd.conf group=sys \
532     original_name=SUNWckr:kernel/drv/sd.conf preserve=true
533 $(i386_ONLY)file path=kernel/drv/sd.conf group=sys \
534     original_name=SUNWos86r:kernel/drv/sd.conf preserve=true
535 $(i386_ONLY)file path=kernel/drv/sgen group=sys
536 file path=kernel/drv/sgen.conf group=sys \
537     original_name=SUNWckr:kernel/drv/sgen.conf preserve=true
538 $(i386_ONLY)file path=kernel/drv/simnet group=sys
539 file path=kernel/drv/simnet.conf group=sys
540 $(i386_ONLY)file path=kernel/drv/smbios group=sys
541 $(i386_ONLY)file path=kernel/drv/smbios.conf group=sys
542 $(i386_ONLY)file path=kernel/drv/softmac group=sys
543 file path=kernel/drv/softmac.conf group=sys
544 $(i386_ONLY)file path=kernel/drv/spdsock group=sys
545 file path=kernel/drv/spdsock.conf group=sys
546 $(i386_ONLY)file path=kernel/drv/st group=sys
547 file path=kernel/drv/st.conf group=sys \
548     original_name=SUNWckr:kernel/drv/st.conf preserve=true
549 $(i386_ONLY)file path=kernel/drv/sy group=sys
550 file path=kernel/drv/sy.conf group=sys
551 $(i386_ONLY)file path=kernel/drv/sysevent group=sys
552 file path=kernel/drv/sysevent.conf group=sys
553 $(i386_ONLY)file path=kernel/drv/sysmsg group=sys
554 file path=kernel/drv/sysmsg.conf group=sys
555 $(i386_ONLY)file path=kernel/drv/tcp group=sys
556 file path=kernel/drv/tcp.conf group=sys
557 $(i386_ONLY)file path=kernel/drv/tcp6 group=sys
558 file path=kernel/drv/tcp6.conf group=sys
559 $(i386_ONLY)file path=kernel/drv/tl group=sys
560 file path=kernel/drv/tl.conf group=sys
561 $(i386_ONLY)file path=kernel/drv/tzmon group=sys
562 $(i386_ONLY)file path=kernel/drv/tzmon.conf group=sys
563 $(sparc_ONLY)file path=kernel/drv/uata.conf group=sys \
564     original_name=SUNWckr:kernel/drv/uata.conf preserve=true
565 $(i386_ONLY)file path=kernel/drv/ucode group=sys
566 $(i386_ONLY)file path=kernel/drv/ucode.conf group=sys
567 $(i386_ONLY)file path=kernel/drv/udp group=sys
568 file path=kernel/drv/udp.conf group=sys
569 $(i386_ONLY)file path=kernel/drv/udp6 group=sys
570 file path=kernel/drv/udp6.conf group=sys
571 $(i386_ONLY)file path=kernel/drv/vgatext group=sys
572 $(i386_ONLY)file path=kernel/drv/vnic group=sys
573 file path=kernel/drv/vnic.conf group=sys
574 $(i386_ONLY)file path=kernel/drv/wc group=sys
575 file path=kernel/drv/wc.conf group=sys
576 $(sparc_ONLY)file path=kernel/exec/$(ARCH64)/aoutexec group=sys mode=0755
577 file path=kernel/exec/$(ARCH64)/elfexec group=sys mode=0755
578 file path=kernel/exec/$(ARCH64)/intpexec group=sys mode=0755
579 $(i386_ONLY)file path=kernel/exec/elfexec group=sys mode=0755
580 $(i386_ONLY)file path=kernel/exec/intpexec group=sys mode=0755
581 file path=kernel/fs/$(ARCH64)/autofs group=sys mode=0755
582 file path=kernel/fs/$(ARCH64)/cachefs group=sys mode=0755
583 file path=kernel/fs/$(ARCH64)/ctfs group=sys mode=0755
584 file path=kernel/fs/$(ARCH64)/dcfs group=sys mode=0755
585 file path=kernel/fs/$(ARCH64)/dev group=sys mode=0755
586 file path=kernel/fs/$(ARCH64)/devfs group=sys mode=0755
587 file path=kernel/fs/$(ARCH64)/fifofs group=sys mode=0755
588 file path=kernel/fs/$(ARCH64)/hsfs group=sys mode=0755
589 file path=kernel/fs/$(ARCH64)/lofs group=sys mode=0755
```



```
590 file path=kernel/fs/$(ARCH64)/mntfs group=sys mode=0755
591 file path=kernel/fs/$(ARCH64)/namefs group=sys mode=0755
592 file path=kernel/fs/$(ARCH64)/objfs group=sys mode=0755
593 file path=kernel/fs/$(ARCH64)/procfs group=sys mode=0755
594 file path=kernel/fs/$(ARCH64)/sharefs group=sys mode=0755
595 file path=kernel/fs/$(ARCH64)/sockfs group=sys mode=0755
596 file path=kernel/fs/$(ARCH64)/specfs group=sys mode=0755
597 file path=kernel/fs/$(ARCH64)/tmpfs group=sys mode=0755
598 file path=kernel/fs/$(ARCH64)/ufs group=sys mode=0755
599 $(i386_ONLY)file path=kernel/fs/autofs group=sys mode=0755
600 $(i386_ONLY)file path=kernel/fs/cachefs group=sys mode=0755
601 $(i386_ONLY)file path=kernel/fs/ctfs group=sys mode=0755
602 $(i386_ONLY)file path=kernel/fs/dcfs group=sys mode=0755
603 $(i386_ONLY)file path=kernel/fs/dev group=sys mode=0755
604 $(i386_ONLY)file path=kernel/fs/devfs group=sys mode=0755
605 $(i386_ONLY)file path=kernel/fs/fifofs group=sys mode=0755
606 $(i386_ONLY)file path=kernel/fs/hsfs group=sys mode=0755
607 $(i386_ONLY)file path=kernel/fs/lofs group=sys mode=0755
608 $(i386_ONLY)file path=kernel/fs/mntfs group=sys mode=0755
609 $(i386_ONLY)file path=kernel/fs/namefs group=sys mode=0755
610 $(i386_ONLY)file path=kernel/fs/objfs group=sys mode=0755
611 $(i386_ONLY)file path=kernel/fs/procfs group=sys mode=0755
612 $(i386_ONLY)file path=kernel/fs/sharefs group=sys mode=0755
613 $(i386_ONLY)file path=kernel/fs/sockfs group=sys mode=0755
614 $(i386_ONLY)file path=kernel/fs/specfs group=sys mode=0755
615 $(i386_ONLY)file path=kernel/fs/tmpfs group=sys mode=0755
616 $(i386_ONLY)file path=kernel/fs/ufs group=sys mode=0755
617 $(i386_ONLY)file path=kernel/genunix group=sys mode=0755
618 file path=kernel/ipp/$(ARCH64)/ipgpc group=sys mode=0755
619 $(i386_ONLY)file path=kernel/ipp/ipgpc group=sys mode=0755
620 file path=kernel/kiconv/$(ARCH64)/kiconv_emea group=sys mode=0755
621 file path=kernel/kiconv/$(ARCH64)/kiconv_ja group=sys mode=0755
622 file path=kernel/kiconv/$(ARCH64)/kiconv_ko group=sys mode=0755
623 file path=kernel/kiconv/$(ARCH64)/kiconv_sc group=sys mode=0755
624 file path=kernel/kiconv/$(ARCH64)/kiconv_tc group=sys mode=0755
625 $(i386_ONLY)file path=kernel/kiconv/kiconv_emea group=sys mode=0755
626 $(i386_ONLY)file path=kernel/kiconv/kiconv_ja group=sys mode=0755
627 $(i386_ONLY)file path=kernel/kiconv/kiconv_ko group=sys mode=0755
628 $(i386_ONLY)file path=kernel/kiconv/kiconv_sc group=sys mode=0755
629 $(i386_ONLY)file path=kernel/kiconv/kiconv_tc group=sys mode=0755
630 file path=kernel/mac/$(ARCH64)/mac_6to4 group=sys mode=0755
631 file path=kernel/mac/$(ARCH64)/mac_ether group=sys mode=0755
632 file path=kernel/mac/$(ARCH64)/mac_ib group=sys mode=0755
633 file path=kernel/mac/$(ARCH64)/mac_ipv4 group=sys mode=0755
634 file path=kernel/mac/$(ARCH64)/mac_ipv6 group=sys mode=0755
635 file path=kernel/mac/$(ARCH64)/mac_wifi group=sys mode=0755
636 $(i386_ONLY)file path=kernel/mac/mac_6to4 group=sys mode=0755
637 $(i386_ONLY)file path=kernel/mac/mac_ether group=sys mode=0755
638 $(i386_ONLY)file path=kernel/mac/mac_ib group=sys mode=0755
639 $(i386_ONLY)file path=kernel/mac/mac_ipv4 group=sys mode=0755
640 $(i386_ONLY)file path=kernel/mac/mac_ipv6 group=sys mode=0755
641 $(i386_ONLY)file path=kernel/mac/mac_wifi group=sys mode=0755
642 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/acpica group=sys mode=0755
643 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/agpmaster group=sys mode=0755
644 file path=kernel/misc/$(ARCH64)/bignum group=sys mode=0755
645 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/bootdev group=sys mode=0755
646 file path=kernel/misc/$(ARCH64)/busra group=sys mode=0755
647 file path=kernel/misc/$(ARCH64)/cardbus group=sys mode=0755
648 file path=kernel/misc/$(ARCH64)/cmlb group=sys mode=0755
649 file path=kernel/misc/$(ARCH64)/consconfig group=sys mode=0755
650 file path=kernel/misc/$(ARCH64)/ctf group=sys mode=0755
651 $(sparc_ONLY)file path=kernel/misc/$(ARCH64)/dada group=sys mode=0755
652 file path=kernel/misc/$(ARCH64)/dls group=sys mode=0755
653 file path=kernel/misc/$(ARCH64)/fssnap_if group=sys mode=0755
654 file path=kernel/misc/$(ARCH64)/gld group=sys mode=0755
655 file path=kernel/misc/$(ARCH64)/hook group=sys mode=0755
```

```
656 file path=kernel/misc/$(ARCH64)/hpcsvc group=sys mode=0755
657 file path=kernel/misc/$(ARCH64)/idmap group=sys mode=0755
658 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/iommulib group=sys mode=0755
659 file path=kernel/misc/$(ARCH64)/ipc group=sys mode=0755
660 file path=kernel/misc/$(ARCH64)/kbtrans group=sys mode=0755
661 file path=kernel/misc/$(ARCH64)/kcf group=sys mode=0755
662 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/kmdbmod group=sys mode=0755
663 file path=kernel/misc/$(ARCH64)/ksocket group=sys mode=0755
664 file path=kernel/misc/$(ARCH64)/mac group=sys mode=0755
665 file path=kernel/misc/$(ARCH64)/mii group=sys mode=0755
666 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/net80211 group=sys mode=0755
667 file path=kernel/misc/$(ARCH64)/neti group=sys mode=0755
668 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/pci_autoconfig group=sys mode=0755
669 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/pcicfg group=sys mode=0755
670 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/pcie group=sys mode=0755
671 file path=kernel/misc/$(ARCH64)/pcihp group=sys mode=0755
672 file path=kernel/misc/$(ARCH64)/pcmcia group=sys mode=0755
673 file path=kernel/misc/$(ARCH64)/rpcsec group=sys mode=0755
674 $(i386_ONLY)file path=kernel/misc/$(ARCH64)/sata group=sys mode=0755
675 file path=kernel/misc/$(ARCH64)/scsi group=sys mode=0755
676 file path=kernel/misc/$(ARCH64)/strplumb group=sys mode=0755
677 $(sparc_ONLY)file path=kernel/misc/$(ARCH64)/swapgeneric group=sys mode=0755
678 file path=kernel/misc/$(ARCH64)/tem group=sys mode=0755
679 file path=kernel/misc/$(ARCH64)/tlimod group=sys mode=0755
680 $(i386_ONLY)file path=kernel/misc/acpica group=sys mode=0755
681 $(i386_ONLY)file path=kernel/misc/agpmaster group=sys mode=0755
682 $(i386_ONLY)file path=kernel/misc/bignum group=sys mode=0755
683 $(i386_ONLY)file path=kernel/misc/bootdev group=sys mode=0755
684 $(i386_ONLY)file path=kernel/misc/busra group=sys mode=0755
685 $(i386_ONLY)file path=kernel/misc/cardbus group=sys mode=0755
686 $(i386_ONLY)file path=kernel/misc/cmlb group=sys mode=0755
687 $(i386_ONLY)file path=kernel/misc/consconfig group=sys mode=0755
688 $(i386_ONLY)file path=kernel/misc/ctf group=sys mode=0755
689 $(i386_ONLY)file path=kernel/misc/dls group=sys mode=0755
690 $(i386_ONLY)file path=kernel/misc/fssnap_if group=sys mode=0755
691 $(i386_ONLY)file path=kernel/misc/gld group=sys mode=0755
692 $(i386_ONLY)file path=kernel/misc/hook group=sys mode=0755
693 $(i386_ONLY)file path=kernel/misc/hpcsvc group=sys mode=0755
694 $(i386_ONLY)file path=kernel/misc/idmap group=sys mode=0755
695 $(i386_ONLY)file path=kernel/misc/iommulib group=sys mode=0755
696 $(i386_ONLY)file path=kernel/misc/ipc group=sys mode=0755
697 $(i386_ONLY)file path=kernel/misc/kbtrans group=sys mode=0755
698 $(i386_ONLY)file path=kernel/misc/kcf group=sys mode=0755
699 $(i386_ONLY)file path=kernel/misc/kmdbmod group=sys mode=0755
700 $(i386_ONLY)file path=kernel/misc/ksocket group=sys mode=0755
701 $(i386_ONLY)file path=kernel/misc/mac group=sys mode=0755
702 $(i386_ONLY)file path=kernel/misc/mii group=sys mode=0755
703 $(i386_ONLY)file path=kernel/misc/net80211 group=sys mode=0755
704 $(i386_ONLY)file path=kernel/misc/neti group=sys mode=0755
705 $(i386_ONLY)file path=kernel/misc/pci_autoconfig group=sys mode=0755
706 $(i386_ONLY)file path=kernel/misc/pcicfg group=sys mode=0755
707 $(i386_ONLY)file path=kernel/misc/pcie group=sys mode=0755
708 $(i386_ONLY)file path=kernel/misc/pcihp group=sys mode=0755
709 $(i386_ONLY)file path=kernel/misc/pcmcia group=sys mode=0755
710 $(i386_ONLY)file path=kernel/misc/rpcsec group=sys mode=0755
711 $(i386_ONLY)file path=kernel/misc/sata group=sys mode=0755
712 $(i386_ONLY)file path=kernel/misc/scsi group=sys mode=0755
713 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_asym_emc group=sys \
714     mode=0755
715 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_asym_lsi group=sys \
716     mode=0755
717 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_asym_sun group=sys \
718     mode=0755
719 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_sym group=sys mode=0755
720 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_sym_emc group=sys \
721     mode=0755
```



```
722 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_sym_hds group=sys \
723     mode=0755
724 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_tape group=sys mode=0755
725 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_tpgs group=sys mode=0755
726 file path=kernel/misc/scsi_vhci/$(ARCH64)/scsi_vhci_f_tpgs_tape group=sys \
727     mode=0755
728 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_asym_emc group=sys \
729     mode=0755
730 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_asym_lsi group=sys \
731     mode=0755
732 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_asym_sun group=sys \
733     mode=0755
734 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_sym group=sys \
735     mode=0755
736 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_sym_emc group=sys \
737     mode=0755
738 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_sym_hds group=sys \
739     mode=0755
740 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_tape group=sys \
741     mode=0755
742 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_tpgs group=sys \
743     mode=0755
744 $(i386_ONLY)file path=kernel/misc/scsi_vhci/scsi_vhci_f_tpgs_tape group=sys \
745     mode=0755
746 $(i386_ONLY)file path=kernel/misc/strplumb group=sys mode=0755
747 $(i386_ONLY)file path=kernel/misc/tem group=sys mode=0755
748 $(i386_ONLY)file path=kernel/misc/tlimod group=sys mode=0755
749 file path=kernel/sched/$(ARCH64)/SDC group=sys mode=0755
750 file path=kernel/sched/$(ARCH64)/TS group=sys mode=0755
751 file path=kernel/sched/$(ARCH64)/TS_DPTBL group=sys mode=0755
752 $(i386_ONLY)file path=kernel/sched/SDC group=sys mode=0755
753 $(i386_ONLY)file path=kernel/sched/TS group=sys mode=0755
754 $(i386_ONLY)file path=kernel/sched/TS_DPTBL group=sys mode=0755
755 file path=kernel/socketmod/$(ARCH64)/ksslf group=sys mode=0755
756 file path=kernel/socketmod/$(ARCH64)/socksctp group=sys mode=0755
757 file path=kernel/socketmod/$(ARCH64)/trill group=sys mode=0755
758 $(i386_ONLY)file path=kernel/socketmod/ksslf group=sys mode=0755
759 $(i386_ONLY)file path=kernel/socketmod/socksctp group=sys mode=0755
760 $(i386_ONLY)file path=kernel/socketmod/trill group=sys mode=0755
761 file path=kernel/strmod/$(ARCH64)/bufmod group=sys mode=0755
762 file path=kernel/strmod/$(ARCH64)/connld group=sys mode=0755
763 file path=kernel/strmod/$(ARCH64)/dedump group=sys mode=0755
764 file path=kernel/strmod/$(ARCH64)/drcompat group=sys mode=0755
765 file path=kernel/strmod/$(ARCH64)/ldterm group=sys mode=0755
766 $(sparc_ONLY)file path=kernel/strmod/$(ARCH64)/ms group=sys mode=0755
767 file path=kernel/strmod/$(ARCH64)/pckt group=sys mode=0755
768 file path=kernel/strmod/$(ARCH64)/pfmod group=sys mode=0755
769 file path=kernel/strmod/$(ARCH64)/pipemod group=sys mode=0755
770 file path=kernel/strmod/$(ARCH64)/ptem group=sys mode=0755
771 file path=kernel/strmod/$(ARCH64)/redirmod group=sys mode=0755
772 file path=kernel/strmod/$(ARCH64)/rpcmod group=sys mode=0755
773 file path=kernel/strmod/$(ARCH64)/timod group=sys mode=0755
774 file path=kernel/strmod/$(ARCH64)/tirdwr group=sys mode=0755
775 file path=kernel/strmod/$(ARCH64)/ttcompat group=sys mode=0755
776 $(sparc_ONLY)file path=kernel/strmod/$(ARCH64)/vuid3ps2 group=sys mode=0755
777 $(i386_ONLY)file path=kernel/strmod/bufmod group=sys mode=0755
778 $(i386_ONLY)file path=kernel/strmod/connld group=sys mode=0755
779 $(i386_ONLY)file path=kernel/strmod/dedump group=sys mode=0755
780 $(i386_ONLY)file path=kernel/strmod/drcompat group=sys mode=0755
781 $(i386_ONLY)file path=kernel/strmod/ldterm group=sys mode=0755
782 $(i386_ONLY)file path=kernel/strmod/pckt group=sys mode=0755
783 $(i386_ONLY)file path=kernel/strmod/pfmod group=sys mode=0755
784 $(i386_ONLY)file path=kernel/strmod/pipemod group=sys mode=0755
785 $(i386_ONLY)file path=kernel/strmod/ptem group=sys mode=0755
786 $(i386_ONLY)file path=kernel/strmod/redirmod group=sys mode=0755
787 $(i386_ONLY)file path=kernel/strmod/rpcmod group=sys mode=0755
```

```
$(i386_ONLY)file path=kernel/strmod/timod group=sys mode=0755
$(i386_ONLY)file path=kernel/strmod/tirdwr group=sys mode=0755
$(i386_ONLY)file path=kernel/strmod/ttcompat group=sys mode=0755
file path=kernel/sys/$(ARCH64)/c2audit group=sys mode=0755
file path=kernel/sys/$(ARCH64)/doorfs group=sys mode=0755
file path=kernel/sys/$(ARCH64)/inst_sync group=sys mode=0755
file path=kernel/sys/$(ARCH64)/kaio group=sys mode=0755
file path=kernel/sys/$(ARCH64)/msgsys group=sys mode=0755
file path=kernel/sys/$(ARCH64)/pipe group=sys mode=0755
file path=kernel/sys/$(ARCH64)/portfs group=sys mode=0755
file path=kernel/sys/$(ARCH64)/pset group=sys mode=0755
file path=kernel/sys/$(ARCH64)/semsys group=sys mode=0755
file path=kernel/sys/$(ARCH64)/shmsys group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/c2audit group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/doorfs group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/inst_sync group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/kaio group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/msgsys group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/pipe group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/portfs group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/pset group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/semsys group=sys mode=0755
$(i386_ONLY)file path=kernel/sys/shmsys group=sys mode=0755
file path=lib/svc/manifest/system/dumpadm.xml group=sys mode=0444
file path=lib/svc/manifest/system/intrd.xml group=sys mode=0444
file path=lib/svc/manifest/system/scheduler.xml group=sys mode=0444
file path=lib/svc/method/svc-dumpadm mode=0555
file path=lib/svc/method/svc-intrd mode=0555
file path=lib/svc/method/svc-scheduler mode=0555
$(sparc_ONLY)file path=usr/share/man/man1m/monitor.1m
$(sparc_ONLY)file path=usr/share/man/man1m/obpsym.1m
# On SPARC driver/bscv is Serverblade1 specific, and in system/kernel/platform
# We keep the manual page generic
$(sparc_ONLY)file path=usr/share/man/man7d/dad.7d
$(i386_ONLY)file path=usr/share/man/man7d/smbios.7d
# Sadly vuid mouse support is in different packages on different platforms
# While kstat(7D) is in SUNWcs, the structures are general
hardlink path=kernel/misc/$(ARCH64)/md5 \
    target=../../../kernel/crypto/$(ARCH64)/md5
hardlink path=kernel/misc/$(ARCH64)/sha1 \
    target=../../../kernel/crypto/$(ARCH64)/sha1
hardlink path=kernel/misc/$(ARCH64)/sha2 \
    target=../../../kernel/crypto/$(ARCH64)/sha2
$(i386_ONLY)hardlink path=kernel/misc/md5 target=../../kernel/crypto/md5
$(i386_ONLY)hardlink path=kernel/misc/sha1 target=../../kernel/crypto/sha1
$(i386_ONLY)hardlink path=kernel/misc/sha2 target=../../kernel/crypto/sha2
hardlink path=kernel/socketmod/$(ARCH64)/icmp \
    target=../../../kernel/drv/$(ARCH64)/icmp
hardlink path=kernel/socketmod/$(ARCH64)/rts \
    target=../../../kernel/drv/$(ARCH64)/rts
hardlink path=kernel/socketmod/$(ARCH64)/tcp \
    target=../../../kernel/drv/$(ARCH64)/tcp
hardlink path=kernel/socketmod/$(ARCH64)/udp \
    target=../../../kernel/drv/$(ARCH64)/udp
$(i386_ONLY)hardlink path=kernel/socketmod/icmp target=../../kernel/drv/icmp
$(i386_ONLY)hardlink path=kernel/socketmod/rts target=../../kernel/drv/rts
$(i386_ONLY)hardlink path=kernel/socketmod/tcp target=../../kernel/drv/tcp
$(i386_ONLY)hardlink path=kernel/socketmod/udp target=../../kernel/drv/udp
hardlink path=kernel/strmod/$(ARCH64)/arp \
    target=../../../kernel/drv/$(ARCH64)/arp
hardlink path=kernel/strmod/$(ARCH64)/icmp \
    target=../../../kernel/drv/$(ARCH64)/icmp
hardlink path=kernel/strmod/$(ARCH64)/ip \
    target=../../../kernel/drv/$(ARCH64)/ip
hardlink path=kernel/strmod/$(ARCH64)/ipsecah \
    target=../../../kernel/drv/$(ARCH64)/ipsecah
```



```
hardlink path=kernel/strmod/$(ARCH64)/ipsecesp \
    target=../../../kernel/drv/$(ARCH64)/ipsecesp
hardlink path=kernel/strmod/$(ARCH64)/keysock \
    target=../../../kernel/drv/$(ARCH64)/keysock
hardlink path=kernel/strmod/$(ARCH64)/tcp \
    target=../../../kernel/drv/$(ARCH64)/tcp
hardlink path=kernel/strmod/$(ARCH64)/udp \
    target=../../../kernel/drv/$(ARCH64)/udp
$(i386_ONLY)hardlink path=kernel/strmod/arp target=../../kernel/drv/arp
$(i386_ONLY)hardlink path=kernel/strmod/icmp target=../../kernel/drv/icmp
$(i386_ONLY)hardlink path=kernel/strmod/ip target=../../kernel/drv/ip
$(i386_ONLY)hardlink path=kernel/strmod/ipsecah \
    target=../../kernel/drv/ipsecah
$(i386_ONLY)hardlink path=kernel/strmod/ipsecesp \
    target=../../kernel/drv/ipsecesp
$(i386_ONLY)hardlink path=kernel/strmod/keysock \
    target=../../kernel/drv/keysock
$(i386_ONLY)hardlink path=kernel/strmod/tcp target=../../kernel/drv/tcp
$(i386_ONLY)hardlink path=kernel/strmod/udp target=../../kernel/drv/udp
hardlink path=kernel/sys/$(ARCH64)/autofs \
    target=../../../kernel/fs/$(ARCH64)/autofs
hardlink path=kernel/sys/$(ARCH64)/rpcmod \
    target=../../../kernel/strmod/$(ARCH64)/rpcmod
$(i386_ONLY)hardlink path=kernel/sys/autofs target=../../kernel/fs/autofs
$(i386_ONLY)hardlink path=kernel/sys/rpcmod target=../../kernel/strmod/rpcmod
legacy pkg=SUNWckr \
    desc="core kernel software for a specific instruction-set architecture" \
    name="Core Solaris Kernel (Root)"
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
license usr/src/cmd/mdb/common/libstand/THIRDPARTYLICENSE \
    license=usr/src/cmd/mdb/common/libstand/THIRDPARTYLICENSE
$(i386_ONLY)license usr/src/common/acpica/THIRDPARTYLICENSE \
    license=usr/src/common/acpica/THIRDPARTYLICENSE
license usr/src/common/bzip2/LICENSE license=usr/src/common/bzip2/LICENSE
license usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams \
    license=usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams
$(i386_ONLY)license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman \
    license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman
$(i386_ONLY)license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl \
    license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl
license usr/src/common/crypto/ecc/THIRDPARTYLICENSE \
    license=usr/src/common/crypto/ecc/THIRDPARTYLICENSE
$(i386_ONLY)license usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE \
    license=usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE
license usr/src/common/mpi/THIRDPARTYLICENSE \
    license=usr/src/common/mpi/THIRDPARTYLICENSE
license usr/src/uts/common/inet/ip/THIRDPARTYLICENSE.rts \
    license=usr/src/uts/common/inet/ip/THIRDPARTYLICENSE.rts
license usr/src/uts/common/inet/tcp/THIRDPARTYLICENSE \
    license=usr/src/uts/common/inet/tcp/THIRDPARTYLICENSE
license usr/src/uts/common/io/THIRDPARTYLICENSE.etheraddr \
    license=usr/src/uts/common/io/THIRDPARTYLICENSE.etheraddr
license usr/src/uts/common/sys/THIRDPARTYLICENSE.icu \
    license=usr/src/uts/common/sys/THIRDPARTYLICENSE.icu
license usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode \
    license=usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode
$(i386_ONLY)license usr/src/uts/intel/io/acpica/THIRDPARTYLICENSE \
    license=usr/src/uts/intel/io/acpica/THIRDPARTYLICENSE
$(i386_ONLY)link path=boot/solaris/bin/root_archive \
    target=../../../usr/sbin/root_archive
link path=dev/dld target=../devices/pseudo/dld@0:ctl
link path=kernel/misc/$(ARCH64)/des \
    target=../../../kernel/crypto/$(ARCH64)/des
$(i386_ONLY)link path=kernel/misc/des target=../../kernel/crypto/des
```

```
**********************************************************
    7434 Thu Dec 26 13:50:15 2013
new/usr/src/tools/findunref/exception_list.open
acpica-unix2-20130823
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# open-tree exception list
#
# See README.exception_lists for details
#

#
# Ignore oddly-named text files scattered about -- someday these should be
# suffixed with .txt so we don't have to list them.
#
./usr/src/cmd/oawk/EXPLAIN
./usr/src/cmd/vi/port/ex.news
./usr/src/cmd/ssh/doc

#
# Ignore everything under trees that may be resynched from outside ON.
#
./usr/src/cmd/perl
./usr/src/cmd/sqlite
./usr/src/lib/libsqlite
./usr/src/cmd/tcpd
./usr/src/common/openssl
./usr/src/grub
./usr/src/cmd/acpitools
./usr/src/common/acpica
./usr/src/uts/intel/sys/acpi

#
# Ignore ksh93/ast-related files that are only used to resync our build
# configuration with upstream.
#
./usr/src/lib/libast/*/src/lib/libast/FEATURE
./usr/src/lib/libast/*/src/lib/libast/ast_namval.h
./usr/src/lib/libast/common/comp/conf.*
./usr/src/lib/libast/common/features
./usr/src/lib/libast/common/include/ast_windows.h
```



```
./usr/src/lib/libast/common/port/lc.tab
./usr/src/lib/libast/common/port/lcgen.c
./usr/src/lib/libcmd/*/src/lib/libcmd/FEATURE
./usr/src/lib/libcmd/common/features
./usr/src/lib/libdll/*/src/lib/libdll/FEATURE
./usr/src/lib/libdll/common/features
./usr/src/lib/libpp/*/pp.*
./usr/src/lib/libpp/common/gentab.sh
./usr/src/lib/libpp/common/ppsym.c
./usr/src/lib/libpp/i386/ppdebug.h
./usr/src/lib/libpp/sparc/ppdebug.h
./usr/src/lib/libshell/*/src/cmd/ksh93/FEATURE
./usr/src/lib/libshell/common/data/math.tab
./usr/src/lib/libshell/common/features
./usr/src/lib/libshell/misc/buildksh93.sh
./usr/src/lib/libshell/misc/buildksh93.readme

#
# Ignore ksh93/ast-related "iffe" (if feature enabled) probe
#
./usr/src/lib/libsum/common/features/sum

#
# Ignore ksh93/ast-related upstream source, currently superseded by
# a per-platform version of sum.h, since we use libmd.so.1 for some
# ciphers.
#
./usr/src/lib/libsum/common/sum.h

#
# Ignore ksh93/ast-related test programs.
#
./usr/src/cmd/ast/msgcc/msgcc.tst
./usr/src/lib/libast/common/port/astmath.c

#
# Ignore ksh93/ast-related source components that are not currently
# used but may be useful later.
#
./usr/src/lib/libcmd/common/cksum.c
./usr/src/lib/libcmd/common/md5sum.c
./usr/src/lib/libcmd/common/sum.c
./usr/src/lib/libshell/common/bltins/mkservice.c
./usr/src/lib/libshell/common/data/bash_pre_rc.sh
./usr/src/lib/libshell/common/include/env.h
./usr/src/lib/libshell/common/sh/bash.c
./usr/src/lib/libshell/common/sh/env.c
./usr/src/lib/libshell/common/sh/shcomp.c
./usr/src/lib/libshell/common/sh/suid_exec.c

#
# Ignore any files built as part of the nightly program itself.
#
# ISUSED - let checkpaths know that the next entry is good.
./usr/src/*.out
# ISUSED - let checkpaths know that the next entry is good.
./usr/src/*.ref

#
# Ignore internal test directories and test programs.
#
*/tests
*/test
*Test.java
*_test.[ch]
./usr/src/cmd/ldap/common/*test.c
```

```
./usr/src/cmd/logadm/tester
./usr/src/cmd/print/printmgr/com/sun/admin/pm/client/helptools/extract
./usr/src/cmd/print/printmgr/com/sun/admin/pm/server/pmtest
./usr/src/cmd/sendmail/libsm/t-*.c
./usr/src/cmd/sort/common/convert.c
./usr/src/cmd/sort/common/invoke.c
./usr/src/lib/crypt_modules/sha256/test.c
./usr/src/lib/efcode/fcode_test
./usr/src/lib/libkvm/common/test.c
./usr/src/lib/libsaveargs/tests/

#
# Ignore debugging code.
#
./usr/src/cmd/fs.d/pcfs/fsck/inject.c
./usr/src/cmd/sort/common/statistics.c

#
# Ignore internal packages, scripts, and tools that are intentionally not
# built or used during a nightly.
#
./usr/src/cmd/sgs/packages
./usr/src/cmd/sgs/rtld.4.x
./usr/src/prototypes
./usr/src/cmd/pools/poold/com/sun/solaris/*/*/package.html
./usr/src/uts/intel/io/acpica/cmp_ca.sh

#
# Ignore files that are only used by internal packages.
#
./usr/src/cmd/sgs/*/*/*chk.msg

#
# Ignore files that get used during a EXPORT_SRC or CRYPT_SRC build only.
#
./usr/src/lib/gss_mechs/mech_dh/backend/mapfile-vers
./usr/src/lib/gss_mechs/mech_dh/dh1024/mapfile-vers
./usr/src/lib/gss_mechs/mech_dh/dh192/mapfile-vers
./usr/src/lib/gss_mechs/mech_dh/dh640/mapfile-vers

#
# Ignore Makefiles which are used by developers but not used by nightly
# itself.  This is a questionable practice, since they tend to rot.
#
./usr/src/cmd/syslogd/sparcv9/Makefile
./usr/src/uts/sparc/uhci/Makefile
./usr/src/lib/pam_modules/smb/amd64/Makefile
./usr/src/lib/pam_modules/smb/sparcv9/Makefile
./usr/src/cmd/isns/isnsd/xml_def/isnsmgmtSchema.xsd

#
# Ignore dtrace scripts only used by developers
#
./usr/src/cmd/vscan/vscand/vscan.d

#
# Ignore sample source code.
#
./usr/src/cmd/sendmail/libmilter/example.c
./usr/src/lib/libdhcpsvc/modules/templates

#
# Ignore .xcl files that aren't used because the program is statically linked.
#
./usr/src/cmd/cmd-inet/sbin/dhcpagent/dhcpagent.xcl
```



```
#
# Ignore sendmail files included for completeness' sake, but which won't
# be used until certain _FFR (for future release) #define's go live.
#
./usr/src/cmd/sendmail/src/statusd_shm.h

#
# Ignore files originally supplied by ISC (Internet Software Consortium)
# as part of a BIND release.
#
./usr/src/lib/libresolv2/common/irs/getaddrinfo.c
./usr/src/lib/libresolv2/common/irs/nis_p.h
./usr/src/lib/libresolv2/common/resolv/res_mkupdate.h
./usr/src/lib/libresolv2/include/err.h

#
# Ignore mont_mulf.c. It is used as a starting point for some hand optimized
# assembly files. We keep it around for future reference.
#
./usr/src/common/bignum/mont_mulf.c

#
# Ignore the sparc Makefiles for x86-only drivers;
# they're used to build warlock only.
#
./usr/src/uts/sparc/sata/Makefile
./usr/src/uts/sparc/si3124/Makefile
./usr/src/uts/sparc/nv_sata/Makefile
./usr/src/uts/sparc/ahci/Makefile

#
# Ignore uttrack.c.  It is provided as part of the standard
# ACPI CA source code but provides optional resource tracking
# functionality which is not used.
#
./usr/src/uts/intel/io/acpica/utilities/uttrack.c

#
# Ignore any files that get used during a gcc build only.
#
./usr/src/cmd/sgs/rtld/common/mapfile-order-gcc

#
# The sharemgr command is built 32-bit only by default, but support
# for building 64-bit is latent in the Makefiles.
#
./usr/src/cmd/dfs.cmds/sharemgr/amd64/Makefile
./usr/src/cmd/dfs.cmds/sharemgr/sparcv9/Makefile

#
# Legitimately unreferenced license/copying files.  Please include
# explanatory comments when adding items here.
#

#
# OPENSOLARIS.LICENSE needs to remain in usr/src as long as it is
# referenced in the CDDL headers.
#
./usr/src/OPENSOLARIS.LICENSE

# Though "COPYING" is usually used as a filename for GPL, the license
# information for openssh is actually found in usr/src/cmd/ssh/doc/LICENSE.
# The COPYING.Ylonen file is merely additional information.
#
./usr/src/cmd/ssh/doc/COPYING.Ylonen
```

```
251 #
252 # This covers header files that are not delivered.
253 #
254 ./usr/src/uts/common/xen/public/COPYING

256 #
257 # ld tests which are not currently delivered
258 #
259 ./usr/src/cmd/sgs/test
260 ./usr/src/cmd/mdb/test
```

```
**********************************************************
   32149 Thu Dec 26 13:50:15 2013
new/usr/src/uts/common/io/power.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2011 Joyent, Inc.  All rights reserved.
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 *      Power Button Driver
 *
 *      This driver handles interrupt generated by the power button on
 *      platforms with "power" device node which has "button" property.
 *      Currently, these platforms are:
 *
 *              ACPI-enabled x86/x64 platforms
 *              Ultra-5_10, Ultra-80, Sun-Blade-100, Sun-Blade-150,
 *              Sun-Blade-1500, Sun-Blade-2500,
 *              Sun-Fire-V210, Sun-Fire-V240, Netra-240
 *
 *      Only one instance is allowed to attach.  In order to know when
 *      an application that has opened the device is going away, a new
 *      minor clone is created for each open(9E) request.  There are
 *      allocations for creating minor clones between 1 and 255.  The ioctl
 *      interface is defined by pbio(7I) and approved as part of
 *      PSARC/1999/393 case.
 */

#include <sys/types.h>
#include <sys/conf.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/cmn_err.h>
#include <sys/errno.h>
#include <sys/modctl.h>
#include <sys/open.h>
#include <sys/stat.h>
#include <sys/poll.h>
#include <sys/pbio.h>
#include <sys/sysevent/eventdefs.h>
#include <sys/sysevent/pwrctl.h>
```



```
#if defined(__sparc)
#include <sys/machsystm.h>
#endif

#ifdef  ACPI_POWER_BUTTON

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

#else

#include <sys/epic.h>
/*
 * Some #defs that must be here as they differ for power.c
 * and epic.c
 */
#define EPIC_REGS_OFFSET        0x00
#define EPIC_REGS_LEN           0x82


/*
 * This flag, which is set for platforms,  that have EPIC processor
 * to process power button interrupt, helps in executing platform
 * specific code.
 */
static char     hasEPIC = B_FALSE;
#endif  /* ACPI_POWER_BUTTON */

/*
 * Maximum number of clone minors that is allowed.  This value
 * is defined relatively low to save memory.
 */
#define POWER_MAX_CLONE 256

/*
 * Minor number is instance << 8 + clone minor from range 1-255; clone 0
 * is reserved for "original" minor.
 */
#define POWER_MINOR_TO_CLONE(minor) ((minor) & (POWER_MAX_CLONE - 1))

/*
 * Power Button Abort Delay
 */
#define ABORT_INCREMENT_DELAY   10

/*
 * FWARC 2005/687: power device compatible property
 */
#define POWER_DEVICE_TYPE "power-device-type"

/*
 * Driver global variables
 */
static void *power_state;
static int power_inst = -1;

static hrtime_t power_button_debounce = NANOSEC/MILLISEC*10;
static hrtime_t power_button_abort_interval = 1.5 * NANOSEC;
static int      power_button_abort_presses = 3;
static int      power_button_abort_enable = 1;
static int      power_button_enable = 1;

static int      power_button_pressed = 0;
static int      power_button_cancel = 0;
```

```
 127 static int      power_button_timeouts = 0;
 128 static int      timeout_cancel = 0;
 129 static int      additional_presses = 0;

 131 /*
 132  * Function prototypes
 133  */
 134 static int power_attach(dev_info_t *, ddi_attach_cmd_t);
 135 static int power_detach(dev_info_t *, ddi_detach_cmd_t);
 136 static int power_getinfo(dev_info_t *, ddi_info_cmd_t, void *, void **);
 137 static int power_open(dev_t *, int, int, cred_t *);
 138 static int power_close(dev_t, int, int, cred_t *);
 139 static int power_ioctl(dev_t, int, intptr_t, int, cred_t *, int *);
 140 static int power_chpoll(dev_t, short, int, short *, struct pollhead **);
 141 #ifndef ACPI_POWER_BUTTON
 142 static uint_t power_high_intr(caddr_t);
 143 #endif
 144 static uint_t power_soft_intr(caddr_t);
 145 static uint_t power_issue_shutdown(caddr_t);
 146 static void power_timeout(caddr_t);
 147 static void power_log_message(void);

 149 /*
 150  * Structure used in the driver
 151  */
 152 struct power_soft_state {
 153         dev_info_t      *dip;           /* device info pointer */
 154         kmutex_t        power_mutex;    /* mutex lock */
 155         kmutex_t        power_intr_mutex; /* interrupt mutex lock */
 156         ddi_iblock_cookie_t soft_iblock_cookie; /* holds interrupt cookie */
 157         ddi_iblock_cookie_t high_iblock_cookie; /* holds interrupt cookie */
 158         ddi_softintr_t  softintr_id;    /* soft interrupt id */
 159         uchar_t         clones[POWER_MAX_CLONE]; /* array of minor clones */
 160         int             monitor_on;     /* clone monitoring the button event */
 161                                         /* clone 0 indicates no one is */
 162                                         /* monitoring the button event */
 163         pollhead_t      pollhd;         /* poll head struct */
 164         int             events;         /* bit map of occured events */
 165         int             shutdown_pending; /* system shutdown in progress */
 166 #ifdef  ACPI_POWER_BUTTON
 167         boolean_t       fixed_attached; /* true means fixed is attached */
 168         boolean_t       gpe_attached;   /* true means GPE is attached */
 169         ACPI_HANDLE     button_obj;     /* handle to device power button */
 170 #else
 171         ddi_acc_handle_t power_rhandle; /* power button register handle */
 172         uint8_t         *power_btn_reg; /* power button register address */
 173         uint8_t         power_btn_bit;  /* power button register bit */
 174         boolean_t       power_regs_mapped; /* flag to tell if regs mapped */
 175         boolean_t       power_btn_ioctl; /* flag to specify ioctl request */
 176 #endif
 177 };
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3541 Thu Dec 26 13:50:16 2013
new/usr/src/uts/common/sys/hotkey_drv.h
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _HOTKEY_DRV_H
#define _HOTKEY_DRV_H

#ifdef  __cplusplus
extern "C" {
#endif

#include <sys/types.h>
#include <sys/conf.h>
#include <sys/stat.h>
#include <sys/note.h>
#include <sys/modctl.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/sysevent/eventdefs.h>
#include <sys/acpi_drv.h>


#define ID_LEN          9

struct acpi_drv_dev {
        ACPI_HANDLE hdl;
        char hid[ID_LEN];       /* ACPI HardwareId */
        char uid[ID_LEN];       /* ACPI UniqueId */
        ACPI_INTEGER adr;       /* Bus device Id */
        int valid;              /* the device state is valid */

        /*
         * Unlike most other devices, when a battery is inserted or
         * removed from the system, the device itself(the battery bay)
         * is still considered to be present in the system.
         *
```



```
         * Value:
         *    0 -- Off-line
         *    1 -- On-line
         *   -1 -- Unknown
         */
        int present;
        enum acpi_drv_type type;
        int index;      /* device index */
        int minor;
};
_____unchanged_portion_omitted_
```

```
**********************************************************
   27764 Thu Dec 26 13:50:16 2013
new/usr/src/uts/i86pc/cpu/amd_opteron/ao_mca.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /*
  23  * Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
  24  */

  26 #include <sys/types.h>
  27 #include <sys/regset.h>
  28 #include <sys/privregs.h>
  29 #include <sys/pci_impl.h>
  30 #include <sys/cpuvar.h>
  31 #include <sys/x86_archext.h>
  32 #include <sys/cmn_err.h>
  33 #include <sys/systm.h>
  34 #include <sys/sysmacros.h>
  35 #include <sys/pghw.h>
  36 #include <sys/cyclic.h>
  37 #include <sys/sysevent.h>
  38 #include <sys/smbios.h>
  39 #include <sys/mca_x86.h>
  40 #include <sys/mca_amd.h>
  41 #include <sys/mc.h>
  42 #include <sys/mc_amd.h>
  43 #include <sys/psw.h>
  44 #include <sys/ddi.h>
  45 #include <sys/sunddi.h>
  46 #include <sys/sdt.h>
  47 #include <sys/fm/util.h>
  48 #include <sys/fm/protocol.h>
  49 #include <sys/fm/cpu/AMD.h>
  50 #include <sys/fm/smb/fmsmb.h>
  51 #include <acpica/include/acpi.h>
  51 #include <sys/acpi/acpi.h>
  52 #include <sys/acpi/acpi_pci.h>
  52 #include <sys/acpica.h>
  53 #include <sys/cpu_module.h>

  55 #include "ao.h"
  56 #include "ao_mca_disp.h"

  58 #define AO_F_REVS_FG (X86_CHIPREV_AMD_F_REV_F | X86_CHIPREV_AMD_F_REV_G)
```



```
  60 int ao_mca_smi_disable = 1;             /* attempt to disable SMI polling */

  62 extern int x86gentopo_legacy;   /* x86 generic topology support */

  64 struct ao_ctl_init {
  65         uint32_t ctl_revmask;   /* rev(s) to which this applies */
  66         uint64_t ctl_bits;      /* mca ctl reg bitmask to set */
  67 };
______unchanged_portion_omitted_
```

```
**********************************************************
    8962 Thu Dec 26 13:50:17 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_container.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * There are three types of container objects defined in the ACPI Spec as below.
 * PNP0A05: Generic Container Device
 *   A device whose settings are totally controlled by its ACPI resource
 *   information, and otherwise needs no device or bus-specific driver support.
 *   This was originally known as Generic ISA Bus Device.
 *   This ID should only be used for containers that do not produce resources
 *   for consumption by child devices. Any system resources claimed by a PNP0A05
 *   device's _CRS object must be consumed by the container itself.
 * PNP0A06: Generic Container Device
 *   This device behaves exactly the same as the PNP0A05 device.
 *   This was originally known as Extended I/O Bus.
 *   This ID should only be used for containers that do not produce resources
 *   for consumption by child devices. Any system resources claimed by a PNP0A06
 *   device's _CRS object must be consumed by the container itself.
 * ACPI0004: Module Device.
 *   This device is a container object that acts as a bus node in a namespace.
 *   A Module Device without any of the _CRS, _PRS and _SRS methods behaves
 *   the same way as the Generic Container Devices (PNP0A05 or PNP0A06).
 *   If the Module Device contains a _CRS method, only the resources
 *   described in the _CRS are available for consumption by its child devices.
 *   Also, the Module Device can support _PRS and _SRS methods if _CRS is
 *   supported.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/note.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
```



```
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_dr.h>
#include <sys/acpidev_impl.h>

static ACPI_STATUS acpidev_container_probe(acpidev_walk_info_t *infop);
static acpidev_filter_result_t acpidev_container_filter(
    acpidev_walk_info_t *infop, char *devname, int maxlen);
static ACPI_STATUS acpidev_container_init(acpidev_walk_info_t *infop);
static acpidev_filter_result_t acpidev_container_filter_func(
    acpidev_walk_info_t *infop, ACPI_HANDLE hdl, acpidev_filter_rule_t *rulep,
    char *devname, int devnamelen);

/*
 * Default class driver for ACPI container objects.
 */
acpidev_class_t acpidev_class_container = {
        0,                              /* adc_refcnt */
        ACPIDEV_CLASS_REV1,             /* adc_version */
        ACPIDEV_CLASS_ID_CONTAINER,     /* adc_class_id */
        "ACPI Container",               /* adc_class_name */
        ACPIDEV_TYPE_CONTAINER,         /* adc_dev_type */
        NULL,                           /* adc_private */
        NULL,                           /* adc_pre_probe */
        NULL,                           /* adc_post_probe */
        acpidev_container_probe,        /* adc_probe */
        acpidev_container_filter,       /* adc_filter */
        acpidev_container_init,         /* adc_init */
        NULL,                           /* adc_fini */
};
______unchanged_portion_omitted_
```

```
**********************************************************
   21688 Thu Dec 26 13:50:17 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_cpu.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * [Support of X2APIC]
 * According to the ACPI Spec, when using the X2APIC interrupt model, logical
 * processors with APIC ID values of 255 and greater are required to have a
 * Processor Device object and must convey the Processor's APIC information to
 * OSPM using the Processor Local X2APIC structure. Logical Processors with APIC
 * ID values less than 255 must use the Processor Local XAPIC structure to
 * convey their APIC information to OSPM.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/bootconf.h>
#include <sys/cpuvar.h>
#include <sys/machsystm.h>
#include <sys/note.h>
#include <sys/psm_types.h>
#include <sys/x86_archext.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_impl.h>

struct acpidev_cpu_map_item {
        uint32_t        proc_id;
        uint32_t        apic_id;
};
______unchanged_portion_omitted_
```

```
**********************************************************
    5107 Thu Dec 26 13:50:18 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_device.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_impl.h>

static ACPI_STATUS acpidev_device_probe(acpidev_walk_info_t *infop);
static acpidev_filter_result_t acpidev_device_filter(acpidev_walk_info_t *infop,
    char *devname, int maxlen);
static ACPI_STATUS acpidev_device_init(acpidev_walk_info_t *infop);

static uint32_t acpidev_device_unitaddr = 0;

/*
 * Default class driver for ACPI DEVICE objects.
 * The default policy for DEVICE objects is to scan child objects without
 * creating device nodes. But some special DEVICE objects will have device
 * nodes created for them.
 */
acpidev_class_t acpidev_class_device = {
        0,                              /* adc_refcnt */
        ACPIDEV_CLASS_REV1,             /* adc_version */
        ACPIDEV_CLASS_ID_DEVICE,        /* adc_class_id */
        "ACPI Device",                  /* adc_class_name */
        ACPIDEV_TYPE_DEVICE,            /* adc_dev_type */
        NULL,                           /* adc_private */
        NULL,                           /* adc_pre_probe */
        NULL,                           /* adc_post_probe */
        acpidev_device_probe,           /* adc_probe */
        acpidev_device_filter,          /* adc_filter */
        acpidev_device_init,            /* adc_init */
        NULL,                           /* adc_fini */
```



```
};
______unchanged_portion_omitted_
```

```
**********************************************************
   73056 Thu Dec 26 13:50:18 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_dr.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/cmn_err.h>
#include <sys/cpuvar.h>
#include <sys/memlist.h>
#include <sys/memlist_impl.h>
#include <sys/note.h>
#include <sys/obpdefs.h>
#include <sys/synch.h>
#include <sys/sysmacros.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <sys/x86_archext.h>
#include <sys/machsystm.h>
#include <sys/memnode.h>        /* for lgrp_plat_node_cnt */
#include <sys/psm_types.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_rsc.h>
#include <sys/acpidev_dr.h>
#include <sys/acpidev_impl.h>

struct acpidev_dr_set_prop_arg {
        uint32_t        level;
        uint32_t        bdnum;
        uint32_t        cpu_id;
        uint32_t        mem_id;
        uint32_t        io_id;
        uint32_t        mod_id;
};
_____unchanged_portion_omitted_
```

```
**********************************************************
   35905 Thu Dec 26 13:50:19 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_drv.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * Platform specific device enumerator for ACPI specific devices.
 * "x86 system devices" refers to the suite of hardware components which are
 * common to the x86 platform and play important roles in the system
 * architecture but can't be enumerated/discovered through industry-standard
 * bus specifications. Examples of these x86 system devices include:
 *   * Logical processor/CPU
 *   * Memory device
 *   * Non-PCI discoverable IOMMU or DMA Remapping Engine
 *   * Non-PCI discoverable IOxAPIC
 *   * Non-PCI discoverable HPET (High Precision Event Timer)
 *   * ACPI defined devices, including power button, sleep button, battery etc.
 *
 * X86 system devices may be discovered through BIOS/Firmware interfaces, such
 * as SMBIOS tables, MPS tables and ACPI tables since their discovery isn't
 * covered by any industry-standard bus specifications.
 *
 * In order to aid Solaris in flexibly managing x86 system devices,
 * x86 system devices are placed into a specific firmware device
 * subtree whose device path is '/devices/fw'.
 *
 * This driver populates the firmware device subtree with ACPI-discoverable
 * system devices if possible. To achieve that, the ACPI object
 * namespace is abstracted as ACPI virtual buses which host system devices.
 * Another nexus driver for the ACPI virtual bus will manage all devices
 * connected to it.
 *
 * For more detailed information, please refer to PSARC/2009/104.
 */

#include <sys/types.h>
#include <sys/bitmap.h>
#include <sys/cmn_err.h>
#include <sys/ddi_subrdefs.h>
#include <sys/errno.h>
#include <sys/modctl.h>
```



```
#include <sys/mutex.h>
#include <sys/note.h>
#include <sys/obpdefs.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_dr.h>
#include <sys/acpidev_impl.h>

/* Patchable through /etc/system */
int acpidev_options = 0;
int acpidev_debug = 0;

krwlock_t acpidev_class_lock;
acpidev_class_list_t *acpidev_class_list_root = NULL;
ulong_t acpidev_object_type_mask[BT_BITOUL(ACPI_TYPE_NS_NODE_MAX + 1)];

/* ACPI device autoconfig global status */
typedef enum acpidev_status {
        ACPIDEV_STATUS_FAILED = -2,     /* ACPI device autoconfig failed */
        ACPIDEV_STATUS_DISABLED = -1,   /* ACPI device autoconfig disabled */
        ACPIDEV_STATUS_UNKNOWN = 0,     /* initial status */
        ACPIDEV_STATUS_INITIALIZED,     /* ACPI device autoconfig initialized */
        ACPIDEV_STATUS_FIRST_PASS,      /* first probing finished */
        ACPIDEV_STATUS_READY            /* second probing finished */
} acpidev_status_t;
```

_____*unchanged_portion_omitted_*

****
5203 Thu Dec 26 13:50:19 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_memory.c
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_rsc.h>
#include <sys/acpidev_dr.h>
#include <sys/acpidev_impl.h>

static ACPI_STATUS acpidev_memory_probe(acpidev_walk_info_t *infop);
static acpidev_filter_result_t acpidev_memory_filter(
    acpidev_walk_info_t *infop, char *devname, int maxlen);
static ACPI_STATUS acpidev_memory_init(acpidev_walk_info_t *infop);

/*
 * Default class driver for ACPI memory objects.
 */
acpidev_class_t acpidev_class_memory = {
        0,                              /* adc_refcnt */
        ACPIDEV_CLASS_REV1,             /* adc_version */
        ACPIDEV_CLASS_ID_MEMORY,        /* adc_class_id */
        "ACPI memory",                  /* adc_class_name */
        ACPIDEV_TYPE_MEMORY,            /* adc_dev_type */
        NULL,                           /* adc_private */
        NULL,                           /* adc_pre_probe */
        NULL,                           /* adc_post_probe */
        acpidev_memory_probe,           /* adc_probe */
        acpidev_memory_filter,          /* adc_filter */
        acpidev_memory_init,            /* adc_init */
        NULL,                           /* adc_fini */
};
```

_____***unchanged_portion_omitted_***

****
4851 Thu Dec 26 13:50:19 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_pci.c
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_rsc.h>
#include <sys/acpidev_dr.h>
#include <sys/acpidev_impl.h>

static ACPI_STATUS acpidev_pci_probe(acpidev_walk_info_t *infop);

/*
 * Default class driver for PCI/PCIEX Host Bridge devices.
 */
acpidev_class_t acpidev_class_pci = {
        0,                              /* adc_refcnt */
        ACPIDEV_CLASS_REV1,             /* adc_version */
        ACPIDEV_CLASS_ID_PCI,           /* adc_class_id */
        "PCI/PCIex Host Bridge",        /* adc_class_name */
        ACPIDEV_TYPE_PCI,               /* adc_dev_type */
        NULL,                           /* adc_private */
        NULL,                           /* adc_pre_probe */
        NULL,                           /* adc_post_probe */
        acpidev_pci_probe,              /* adc_probe */
        NULL,                           /* adc_filter */
        NULL,                           /* adc_init */
        NULL,                           /* adc_fini */
};
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   30644 Thu Dec 26 13:50:20 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_resource.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/cmn_err.h>
#include <sys/sysmacros.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_rsc.h>
#include <sys/acpidev_impl.h>

#define ACPIDEV_RES_INIT_ITEMS          8
#define ACPIDEV_RES_INCR_ITEMS          8

/* Data structure to hold parsed resources during walking. */
struct acpidev_resource_handle {
        boolean_t                       acpidev_consumer;
        int                             acpidev_reg_count;
        int                             acpidev_reg_max;
        acpidev_phys_spec_t             *acpidev_regp;
        acpidev_phys_spec_t             acpidev_regs[ACPIDEV_RES_INIT_ITEMS];
        int                             acpidev_range_count;
        int                             acpidev_range_max;
        acpidev_ranges_t                *acpidev_rangep;
        acpidev_ranges_t                acpidev_ranges[ACPIDEV_RES_INIT_ITEMS];
        int                             acpidev_bus_count;
        int                             acpidev_bus_max;
        acpidev_bus_range_t             *acpidev_busp;
        acpidev_bus_range_t             acpidev_buses[ACPIDEV_RES_INIT_ITEMS];
        int                             acpidev_irq_count;
        int                             acpidev_irqp[ACPIDEV_RES_IRQ_MAX];
```



```
        int                             acpidev_dma_count;
        int                             acpidev_dmap[ACPIDEV_RES_DMA_MAX];
};
______unchanged_portion_omitted_
```

****
5148 Thu Dec 26 13:50:20 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_scope.c
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_impl.h>

static ACPI_STATUS acpidev_scope_probe(acpidev_walk_info_t *infop);
static acpidev_filter_result_t acpidev_scope_filter(acpidev_walk_info_t *infop,
    char *devname, int maxlen);
static ACPI_STATUS acpidev_scope_init(acpidev_walk_info_t *infop);

/*
 * Default class driver for ACPI scope objects.
 * This class driver is used to handle predefined ACPI SCOPE objects
 * under the ACPI root object, such as _PR_, _SB_ and _TZ_ etc.
 * The default policy for ACPI SCOPE objects is SKIP.
 */
acpidev_class_t acpidev_class_scope = {
        0,                              /* adc_refcnt */
        ACPIDEV_CLASS_REV1,             /* adc_version */
        ACPIDEV_CLASS_ID_SCOPE,         /* adc_class_id */
        "ACPI Scope",                   /* adc_class_name */
        ACPIDEV_TYPE_SCOPE,             /* adc_dev_type */
        NULL,                           /* adc_private */
        NULL,                           /* adc_pre_probe */
        NULL,                           /* adc_post_probe */
        acpidev_scope_probe,            /* adc_probe */
        acpidev_scope_filter,           /* adc_filter */
        acpidev_scope_init,             /* adc_init */
        NULL,                           /* adc_fini */
};
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   24027 Thu Dec 26 13:50:21 2013
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_util.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/cmn_err.h>
#include <sys/note.h>
#include <sys/sysmacros.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_impl.h>
#include <util/sscanf.h>

/* Data structures used to extract the numeric unit address from string _UID. */
static acpidev_pseudo_uid_head_t acpidev_uid_heads[ACPIDEV_CLASS_ID_MAX];
static char *acpidev_uid_formats[] = {
        "%u",
};
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   21526 Thu Dec 26 13:50:21 2013
new/usr/src/uts/i86pc/io/acpi/acpinex/acpinex_event.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/atomic.h>
#include <sys/bitmap.h>
#include <sys/cmn_err.h>
#include <sys/note.h>
#include <sys/sunndi.h>
#include <sys/fastboot_impl.h>
#include <sys/sysevent.h>
#include <sys/sysevent/dr.h>
#include <sys/sysevent/eventdefs.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_dr.h>
#include <sys/acpinex.h>

int acpinex_event_support_remove = 0;

static volatile uint_t acpinex_dr_event_cnt = 0;
static ulong_t acpinex_object_type_mask[BT_BITOUL(ACPI_TYPE_NS_NODE_MAX + 1)];

/*
 * Generate DR_REQ event to syseventd.
 * Please refer to sys/sysevent/dr.h for message definition.
 */
static int
acpinex_event_generate_event(dev_info_t *dip, ACPI_HANDLE hdl, int req,
    int event, char *objname)
{
        int rv = 0;
        sysevent_id_t eid;
        sysevent_value_t evnt_val;
        sysevent_attr_list_t *evnt_attr_list = NULL;
```



```
        char *attach_pnt;
        char event_type[32];

        /* Add "attachment point" attribute. */
        attach_pnt = kmem_zalloc(MAXPATHLEN, KM_SLEEP);
        if (ACPI_FAILURE(acpidev_dr_get_attachment_point(hdl,
            attach_pnt, MAXPATHLEN))) {
                cmn_err(CE_WARN,
                    "!acpinex: failed to generate AP name for %s.", objname);
                kmem_free(attach_pnt, MAXPATHLEN);
                return (-1);
        }
        ASSERT(attach_pnt[0] != '\0');
        evnt_val.value_type = SE_DATA_TYPE_STRING;
        evnt_val.value.sv_string = attach_pnt;
        rv = sysevent_add_attr(&evnt_attr_list, DR_AP_ID, &evnt_val, KM_SLEEP);
        if (rv != 0) {
                cmn_err(CE_WARN,
                    "!acpinex: failed to add attr [%s] for %s event.",
                    DR_AP_ID, EC_DR);
                kmem_free(attach_pnt, MAXPATHLEN);
                return (rv);
        }

        /* Add "request type" attribute. */
        evnt_val.value_type = SE_DATA_TYPE_STRING;
        evnt_val.value.sv_string = SE_REQ2STR(req);
        rv = sysevent_add_attr(&evnt_attr_list, DR_REQ_TYPE, &evnt_val,
            KM_SLEEP);
        if (rv != 0) {
                cmn_err(CE_WARN,
                    "!acpinex: failed to add attr [%s] for %s event.",
                    DR_REQ_TYPE, EC_DR);
                sysevent_free_attr(evnt_attr_list);
                kmem_free(attach_pnt, MAXPATHLEN);
                return (rv);
        }

        /* Add "acpi-event-type" attribute. */
        switch (event) {
        case ACPI_NOTIFY_BUS_CHECK:
                (void) snprintf(event_type, sizeof (event_type),
                    ACPIDEV_EVENT_TYPE_BUS_CHECK);
                break;
        case ACPI_NOTIFY_DEVICE_CHECK:
                (void) snprintf(event_type, sizeof (event_type),
                    ACPIDEV_EVENT_TYPE_DEVICE_CHECK);
                break;
        case ACPI_NOTIFY_DEVICE_CHECK_LIGHT:
                (void) snprintf(event_type, sizeof (event_type),
                    ACPIDEV_EVENT_TYPE_DEVICE_CHECK_LIGHT);
                break;
        case ACPI_NOTIFY_EJECT_REQUEST:
                (void) snprintf(event_type, sizeof (event_type),
                    ACPIDEV_EVENT_TYPE_EJECT_REQUEST);
                break;
        default:
                cmn_err(CE_WARN,
                    "!acpinex: unknown ACPI event type %d.", event);
                sysevent_free_attr(evnt_attr_list);
                kmem_free(attach_pnt, MAXPATHLEN);
                return (-1);
        }
        evnt_val.value_type = SE_DATA_TYPE_STRING;
        evnt_val.value.sv_string = event_type;
        rv = sysevent_add_attr(&evnt_attr_list, ACPIDEV_EVENT_TYPE_ATTR_NAME,
```

```
 127             &evnt_val, KM_SLEEP);
 128         if (rv != 0) {
 129                 cmn_err(CE_WARN,
 130                     "!acpinex: failed to add attr [%s] for %s event.",
 131                     ACPIDEV_EVENT_TYPE_ATTR_NAME, EC_DR);
 132                 sysevent_free_attr(evnt_attr_list);
 133                 kmem_free(attach_pnt, MAXPATHLEN);
 134                 return (rv);
 135         }

 137         rv = ddi_log_sysevent(dip, DDI_VENDOR_SUNW, EC_DR, ESC_DR_REQ,
 138             evnt_attr_list, &eid, KM_SLEEP);
 139         if (rv != DDI_SUCCESS) {
 140                 cmn_err(CE_WARN,
 141                     "!acpinex: failed to log DR_REQ event for %s.", objname);
 142                 rv = -1;
 143         }

 145         nvlist_free(evnt_attr_list);
 146         kmem_free(attach_pnt, MAXPATHLEN);

 148         return (rv);
 149 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   66613 Thu Dec 26 13:50:22 2013
new/usr/src/uts/i86pc/io/acpi/drmach_acpi/drmach_acpi.c
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */
  25 /*
  26  * Copyright (c) 2010, Intel Corporation.
  27  * All rights reserved.
  28  */

  30 #include <sys/types.h>
  31 #include <sys/cmn_err.h>
  32 #include <sys/conf.h>
  33 #include <sys/debug.h>
  34 #include <sys/errno.h>
  35 #include <sys/note.h>
  36 #include <sys/dditypes.h>
  37 #include <sys/ddi.h>
  38 #include <sys/sunddi.h>
  39 #include <sys/sunndi.h>
  40 #include <sys/ddi_impldefs.h>
  41 #include <sys/ndi_impldefs.h>
  42 #include <sys/varargs.h>
  43 #include <sys/modctl.h>
  44 #include <sys/kmem.h>
  45 #include <sys/cpuvar.h>
  46 #include <sys/cpupart.h>
  47 #include <sys/mem_config.h>
  48 #include <sys/mem_cage.h>
  49 #include <sys/memnode.h>
  50 #include <sys/callb.h>
  51 #include <sys/ontrap.h>
  52 #include <sys/obpdefs.h>
  53 #include <sys/promif.h>
  54 #include <sys/synch.h>
  55 #include <sys/systm.h>
  56 #include <sys/sysmacros.h>
  57 #include <sys/archsystm.h>
  58 #include <sys/machsystm.h>
  59 #include <sys/x_call.h>
  60 #include <sys/x86_archext.h>
  61 #include <sys/fastboot_impl.h>
```



```
  62 #include <sys/sysevent.h>
  63 #include <sys/sysevent/dr.h>
  64 #include <sys/sysevent/eventdefs.h>
  65 #include <acpica/include/acpi.h>
  65 #include <sys/acpi/acpi.h>
  66 #include <sys/acpica.h>
  67 #include <sys/acpidev.h>
  68 #include <sys/acpidev_rsc.h>
  69 #include <sys/acpidev_dr.h>
  70 #include <sys/dr.h>
  71 #include <sys/dr_util.h>
  72 #include <sys/drmach.h>
  73 #include "drmach_acpi.h"

  75 /* utility */
  76 #define MBYTE           (1048576ull)
  77 #define _ptob64(p)      ((uint64_t)(p) << PAGESHIFT)
  78 #define _b64top(b)      ((pgcnt_t)((b) >> PAGESHIFT))

  80 static int              drmach_init(void);
  81 static void             drmach_fini(void);
  82 static int              drmach_name2type_idx(char *);
  83 static sbd_error_t      *drmach_mem_update_lgrp(drmachid_t);

  85 static void drmach_board_dispose(drmachid_t id);
  86 static sbd_error_t *drmach_board_release(drmachid_t);
  87 static sbd_error_t *drmach_board_status(drmachid_t, drmach_status_t *);

  89 static void drmach_io_dispose(drmachid_t);
  90 static sbd_error_t *drmach_io_release(drmachid_t);
  91 static sbd_error_t *drmach_io_status(drmachid_t, drmach_status_t *);

  93 static void drmach_cpu_dispose(drmachid_t);
  94 static sbd_error_t *drmach_cpu_release(drmachid_t);
  95 static sbd_error_t *drmach_cpu_status(drmachid_t, drmach_status_t *);

  97 static void drmach_mem_dispose(drmachid_t);
  98 static sbd_error_t *drmach_mem_release(drmachid_t);
  99 static sbd_error_t *drmach_mem_status(drmachid_t, drmach_status_t *);

 101 #ifdef DEBUG
 102 int drmach_debug = 1;           /* set to non-zero to enable debug messages */
 103 #endif /* DEBUG */

 105 drmach_domain_info_t     drmach_domain;

 107 static char             *drmach_ie_fmt = "drmach_acpi.c %d";
 108 static drmach_array_t   *drmach_boards;

 110 /* rwlock to protect drmach_boards. */
 111 static krwlock_t         drmach_boards_rwlock;

 113 /* rwlock to block out CPR thread. */
 114 static krwlock_t         drmach_cpr_rwlock;

 116 /* CPR callb id. */
 117 static callb_id_t        drmach_cpr_cid;

 119 static struct {
 120         const char      *name;
 121         const char      *type;
 122         sbd_error_t     *(*new)(drmach_device_t *, drmachid_t *);
 123 } drmach_name2type[] = {
______unchanged_portion_omitted_
```

```
**********************************************************
    5136 Thu Dec 26 13:50:22 2013
new/usr/src/uts/i86pc/io/acpi/drmach_acpi/drmach_acpi.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#ifndef _SYS_DRMACH_ACPI_H
#define _SYS_DRMACH_ACPI_H
#include <sys/types.h>
#include <sys/cmn_err.h>
#include <sys/param.h>
#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/drmach.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifdef  _KERNEL

/* Use ACPI handle as DRMACH handle on x86 systems. */
#define DRMACH_HANDLE           ACPI_HANDLE

/* Macros to deal with object type. */
#define DRMACH_OBJ(id)          ((drmach_common_t *)id)

#define DRMACH_NULL_ID(id)      ((id) == 0)

#define DRMACH_IS_BOARD_ID(id)  \
        ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_board_new))

#define DRMACH_IS_CPU_ID(id)    \
        ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new))
```



```
#define DRMACH_IS_MEM_ID(id)    \
        ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_mem_new))

#define DRMACH_IS_IO_ID(id)     \
        ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

#define DRMACH_IS_DEVICE_ID(id)                                 \
        ((id != 0) &&                                           \
        (DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new ||       \
            DRMACH_OBJ(id)->isa == (void *)drmach_mem_new ||    \
            DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

#define DRMACH_IS_ID(id)                                        \
        ((id != 0) &&                                           \
        (DRMACH_OBJ(id)->isa == (void *)drmach_board_new ||     \
            DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new ||    \
            DRMACH_OBJ(id)->isa == (void *)drmach_mem_new ||    \
            DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

#define DRMACH_INTERNAL_ERROR() \
        drerr_new(1, EX86_INTERNAL, drmach_ie_fmt, __LINE__)

#ifdef DEBUG
extern int drmach_debug;

#define DRMACH_PR               if (drmach_debug) printf
#else
#define DRMACH_PR               _NOTE(CONSTANTCONDITION) if (0) printf
#endif /* DEBUG */

typedef struct {
        struct drmach_node      *node;
        void                    *data;
        void                    *func;
} drmach_node_walk_args_t;
```

*_____unchanged_portion_omitted_*

```
**********************************************************
    7428 Thu Dec 26 13:50:23 2013
new/usr/src/uts/i86pc/io/amd_iommu/amd_iommu_acpi.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#ifndef _AMD_IOMMU_ACPI_H
#define _AMD_IOMMU_ACPI_H

#ifdef __cplusplus
extern "C" {
#endif

#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpica.h>
#include <sys/amd_iommu.h>
#include "amd_iommu_impl.h"

#ifdef _KERNEL

#define IVRS_SIG        "IVRS"

/*
 * IVINFO settings
 */
#define AMD_IOMMU_ACPI_IVINFO_RSV1      (31 << 16 | 23)
#define AMD_IOMMU_ACPI_HT_ATSRSV        (22 << 16 | 22)
#define AMD_IOMMU_ACPI_VA_SIZE          (21 << 16 | 15)
#define AMD_IOMMU_ACPI_PA_SIZE          (14 << 16 | 8)
#define AMD_IOMMU_ACPI_IVINFO_RSV2      (7 << 16 | 0)

/*
 * IVHD Device entry len field
 */
#define AMD_IOMMU_ACPI_DEVENTRY_LEN     (7 << 16 | 6)

/*
 * IVHD flag fields definition
 */
#define AMD_IOMMU_ACPI_IVHD_FLAGS_RSV           (7 << 16 | 5)
#define AMD_IOMMU_ACPI_IVHD_FLAGS_IOTLBSUP      (4 << 16 | 4)
```



```
#define AMD_IOMMU_ACPI_IVHD_FLAGS_ISOC          (3 << 16 | 3)
#define AMD_IOMMU_ACPI_IVHD_FLAGS_RESPASSPW     (2 << 16 | 2)
#define AMD_IOMMU_ACPI_IVHD_FLAGS_PASSPW        (1 << 16 | 1)
#define AMD_IOMMU_ACPI_IVHD_FLAGS_HTTUNEN       (0 << 16 | 0)

/*
 * IVHD IOMMU info fields
 */
#define AMD_IOMMU_ACPI_IOMMU_INFO_RSV1          (15 << 16 | 13)
#define AMD_IOMMU_ACPI_IOMMU_INFO_UNITID        (12 << 16 | 8)
#define AMD_IOMMU_ACPI_IOMMU_INFO_RSV2          (7 << 16 | 5)
#define AMD_IOMMU_ACPI_IOMMU_INFO_MSINUM        (4 << 16 | 0)

/*
 * IVHD deventry data settings
 */
#define AMD_IOMMU_ACPI_LINT1PASS        (7 << 16 | 7)
#define AMD_IOMMU_ACPI_LINT0PASS        (6 << 16 | 6)
#define AMD_IOMMU_ACPI_SYSMGT           (5 << 16 | 4)
#define AMD_IOMMU_ACPI_DATRSV           (3 << 16 | 3)
#define AMD_IOMMU_ACPI_NMIPASS          (2 << 16 | 2)
#define AMD_IOMMU_ACPI_EXTINTPASS       (1 << 16 | 1)
#define AMD_IOMMU_ACPI_INITPASS         (0 << 16 | 0)

/*
 * IVHD deventry extended data settings
 */
#define AMD_IOMMU_ACPI_ATSDISABLED      (31 << 16 | 31)
#define AMD_IOMMU_ACPI_EXTDATRSV        (30 << 16 | 0)

/*
 * IVMD flags fields settings
 */
#define AMD_IOMMU_ACPI_IVMD_RSV         (7 << 16 | 4)
#define AMD_IOMMU_ACPI_IVMD_EXCL_RANGE  (3 << 16 | 3)
#define AMD_IOMMU_ACPI_IVMD_IW          (2 << 16 | 2)
#define AMD_IOMMU_ACPI_IVMD_IR          (1 << 16 | 1)
#define AMD_IOMMU_ACPI_IVMD_UNITY       (0 << 16 | 0)

#define AMD_IOMMU_ACPI_INFO_HASH_SZ     (256)

/*
 * Deventry special device "variety"
 */
#define AMD_IOMMU_ACPI_SPECIAL_APIC     0x1
#define AMD_IOMMU_ACPI_SPECIAL_HPET     0x2

typedef enum {
        DEVENTRY_INVALID = 0,
        DEVENTRY_ALL = 1,
        DEVENTRY_SELECT,
        DEVENTRY_RANGE,
        DEVENTRY_RANGE_END,
        DEVENTRY_ALIAS_SELECT,
        DEVENTRY_ALIAS_RANGE,
        DEVENTRY_EXTENDED_SELECT,
        DEVENTRY_EXTENDED_RANGE,
        DEVENTRY_SPECIAL_DEVICE
} ivhd_deventry_type_t;
```

*unchanged_portion_omitted_*

```
**********************************************************
   45674 Thu Dec 26 13:50:23 2013
new/usr/src/uts/i86pc/io/amd_iommu/amd_iommu_page_tables.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/amd_iommu.h>
#include <sys/bootconf.h>
#include <sys/sysmacros.h>
#include <sys/ddidmareq.h>

#include "amd_iommu_impl.h"
#include "amd_iommu_acpi.h"
#include "amd_iommu_page_tables.h"

ddi_dma_attr_t amd_iommu_pgtable_dma_attr = {
        DMA_ATTR_V0,
        0U,                             /* dma_attr_addr_lo */
        0xffffffffffffffffULL,          /* dma_attr_addr_hi */
        0xffffffffU,                    /* dma_attr_count_max */
        (uint64_t)4096,                 /* dma_attr_align */
        1,                              /* dma_attr_burstsizes */
        64,                             /* dma_attr_minxfer */
        0xffffffffU,                    /* dma_attr_maxxfer */
        0xffffffffU,                    /* dma_attr_seg */
        1,                              /* dma_attr_sgllen, variable */
        64,                             /* dma_attr_granular */
        0                               /* dma_attr_flags */
};
______unchanged_portion_omitted_
```

```
**********************************************************
   65839 Thu Dec 26 13:50:24 2013
new/usr/src/uts/i86pc/io/apix/apix.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright (c) 2013, Joyent, Inc.  All rights reserved.
 */

/*
 * To understand how the apix module interacts with the interrupt subsystem read
 * the theory statement in uts/i86pc/os/intr.c.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
```



```
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/cyclic.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/mach_intr.h>
#include <sys/apix.h>
#include <sys/apix_irm_impl.h>

static int apix_probe();
static void apix_init();
static void apix_picinit(void);
static int apix_intr_enter(int, int *);
static void apix_intr_exit(int, int);
static void apix_setspl(int);
static int apix_disable_intr(processorid_t);
static void apix_enable_intr(processorid_t);
static int apix_get_clkvect(int);
static int apix_get_ipivect(int, int);
static void apix_post_cyclic_setup(void *);
static int apix_post_cpu_start();
static int apix_intr_ops(dev_info_t *, ddi_intr_handle_impl_t *,
    psm_intr_op_t, int *);

/*
 * Helper functions for apix_intr_ops()
 */
static void apix_redistribute_compute(void);
static int apix_get_pending(apix_vector_t *);
static apix_vector_t *apix_get_req_vector(ddi_intr_handle_impl_t *, ushort_t);
static int apix_get_intr_info(ddi_intr_handle_impl_t *, apic_get_intr_t *);
static char *apix_get_apic_type(void);
static int apix_intx_get_pending(int);
static void apix_intx_set_mask(int irqno);
static void apix_intx_clear_mask(int irqno);
static int apix_intx_get_shared(int irqno);
static void apix_intx_set_shared(int irqno, int delta);
static apix_vector_t *apix_intx_xlate_vector(dev_info_t *, int,
    struct intrspec *);
static int apix_intx_alloc_vector(dev_info_t *, int, struct intrspec *);

extern int apic_clkinit(int);

/* IRM initialization for APIX PSM module */
extern void apix_irm_init(void);

extern int irm_enable;

/*
```

```
 127  *      Local static data
 128  */
 129 static struct   psm_ops apix_ops = {
 130         apix_probe,

 132         apix_init,
 133         apix_picinit,
 134         apix_intr_enter,
 135         apix_intr_exit,
 136         apix_setspl,
 137         apix_addspl,
 138         apix_delspl,
 139         apix_disable_intr,
 140         apix_enable_intr,
 141         NULL,                   /* psm_softlvl_to_irq */
 142         NULL,                   /* psm_set_softintr */

 144         apic_set_idlecpu,
 145         apic_unset_idlecpu,

 147         apic_clkinit,
 148         apix_get_clkvect,
 149         NULL,                   /* psm_hrtimeinit */
 150         apic_gethrtime,

 152         apic_get_next_processorid,
 153         apic_cpu_start,
 154         apix_post_cpu_start,
 155         apic_shutdown,
 156         apix_get_ipivect,
 157         apic_send_ipi,

 159         NULL,                   /* psm_translate_irq */
 160         NULL,                   /* psm_notify_error */
 161         NULL,                   /* psm_notify_func */
 162         apic_timer_reprogram,
 163         apic_timer_enable,
 164         apic_timer_disable,
 165         apix_post_cyclic_setup,
 166         apic_preshutdown,
 167         apix_intr_ops,          /* Advanced DDI Interrupt framework */
 168         apic_state,             /* save, restore apic state for S3 */
 169         apic_cpu_ops,           /* CPU control interface. */
 170 };
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   48694 Thu Dec 26 13:50:24 2013
new/usr/src/uts/i86pc/io/apix/apix_utils.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 Pluribus Networks, Inc.
 */

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
```



```
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/apix.h>

static int apix_get_avail_vector_oncpu(uint32_t, int, int);
static apix_vector_t *apix_init_vector(processorid_t, uchar_t);
static void apix_cleanup_vector(apix_vector_t *);
static void apix_insert_av(apix_vector_t *, void *, avfunc, caddr_t, caddr_t,
    uint64_t *, int, dev_info_t *);
static void apix_remove_av(apix_vector_t *, struct autovec *);
static void apix_clear_dev_map(dev_info_t *, int, int);
static boolean_t apix_is_cpu_enabled(processorid_t);
static void apix_wait_till_seen(processorid_t, int);

#define GET_INTR_INUM(ihdlp)            \
        (((ihdlp) != NULL) ? ((ddi_intr_handle_impl_t *)(ihdlp))->ih_inum : 0)

apix_rebind_info_t apix_rebindinfo = {0, 0, 0, NULL, 0, NULL};

/*
 * Allocate IPI
 *
 * Return vector number or 0 on error
 */
uchar_t
apix_alloc_ipi(int ipl)
{
        apix_vector_t *vecp;
        uchar_t vector;
        int cpun;
        int nproc;

        APIX_ENTER_CPU_LOCK(0);

        vector = apix_get_avail_vector_oncpu(0, APIX_IPI_MIN, APIX_IPI_MAX);
        if (vector == 0) {
                APIX_LEAVE_CPU_LOCK(0);
                cmn_err(CE_WARN, "apix: no available IPI\n");
                apic_error |= APIC_ERR_GET_IPIVECT_FAIL;
                return (0);
        }

        nproc = max(apic_nproc, apic_max_nproc);
        for (cpun = 0; cpun < nproc; cpun++) {
                vecp = xv_vector(cpun, vector);
                if (vecp == NULL) {
                        vecp = kmem_zalloc(sizeof (apix_vector_t), KM_NOSLEEP);
                        if (vecp == NULL) {
                                cmn_err(CE_WARN, "apix: No memory for ipi");
                                goto fail;
                        }
                        xv_vector(cpun, vector) = vecp;
                }
                vecp->v_state = APIX_STATE_ALLOCED;
                vecp->v_type = APIX_TYPE_IPI;
                vecp->v_cpuid = vecp->v_bound_cpuid = cpun;
                vecp->v_vector = vector;
                vecp->v_pri = ipl;
        }
        APIX_LEAVE_CPU_LOCK(0);
        return (vector);
```

```
 128 fail:
 129         while (--cpun >= 0)
 130                 apix_cleanup_vector(xv_vector(cpun, vector));
 131         APIX_LEAVE_CPU_LOCK(0);
 132         return (0);
 133 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   27729 Thu Dec 26 13:50:25 2013
new/usr/src/uts/i86pc/io/immu_dmar.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * Copyright (c) 2009, Intel Corporation.
 * All rights reserved.
 */


#include <sys/debug.h>
#include <sys/sysmacros.h>
#include <sys/types.h>
#include <sys/kmem.h>
#include <sys/sunddi.h>
#include <sys/list.h>
#include <sys/pci.h>
#include <sys/pci_cfgspace.h>
#include <sys/pci_impl.h>
#include <sys/sunndi.h>
#include <sys/ksynch.h>
#include <sys/cmn_err.h>
#include <sys/bootconf.h>
#include <sys/int_fmtio.h>
#include <sys/smbios.h>
#include <sys/apic.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/immu.h>
#include <sys/smp_impldefs.h>

static void dmar_table_destroy(dmar_table_t *tbl);

/*
 * internal global variables
 */
static char     *dmar_raw;              /* raw DMAR ACPI table */
static dmar_table_t *dmar_table;        /* converted form of DMAR table */

/*
```



```
 * global variables exported outside this file
 */
boolean_t dmar_print = B_FALSE;
kmutex_t ioapic_drhd_lock;
list_t ioapic_drhd_list;

/* ########################################################################### */

/*
 * helper functions to read the "raw" DMAR table
 */

static uint8_t
get_uint8(char *cp)
{
        uint8_t val = *((uint8_t *)cp);
        return (val);
}
_____unchanged_portion_omitted_
```

```
**********************************************************
   78638 Thu Dec 26 13:50:25 2013
new/usr/src/uts/i86pc/io/immu_dvma.c
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Portions Copyright (c) 2010, Oracle and/or its affiliates.
  23  * All rights reserved.
  24  */
  25 /*
  26  * Copyright (c) 2009, Intel Corporation.
  27  * All rights reserved.
  28  */
  29 /*
  30  * Copyright 2012 Garrett D'Amore <garrett@damore.org>.  All rights reserved.
  31  */

  33 /*
  34  * DVMA code
  35  * This file contains Intel IOMMU code that deals with DVMA
  36  * i.e. DMA remapping.
  37  */

  39 #include <sys/sysmacros.h>
  40 #include <sys/pcie.h>
  41 #include <sys/pci_cfgspace.h>
  42 #include <vm/hat_i86.h>
  43 #include <sys/memlist.h>
  44 #include <acpica/include/acpi.h>
  44 #include <sys/acpi/acpi.h>
  45 #include <sys/acpica.h>
  46 #include <sys/modhash.h>
  47 #include <sys/immu.h>
  48 #include <sys/x86_archext.h>
  49 #include <sys/archsystm.h>

  51 #undef  TEST

  53 /*
  54  * Macros based on PCI spec
  55  */
  56 #define IMMU_PCI_REV2CLASS(r)   ((r) >> 8)  /* classcode from revid */
  57 #define IMMU_PCI_CLASS2BASE(c)  ((c) >> 16) /* baseclass from classcode */
  58 #define IMMU_PCI_CLASS2SUB(c)   (((c) >> 8) & 0xff); /* classcode */

  60 #define IMMU_CONTIG_PADDR(d, p) \
```



```
  61         ((d).dck_paddr && ((d).dck_paddr + IMMU_PAGESIZE) == (p))

  63 typedef struct dvma_arg {
  64         immu_t *dva_immu;
  65         dev_info_t *dva_rdip;
  66         dev_info_t *dva_ddip;
  67         domain_t *dva_domain;
  68         int dva_level;
  69         immu_flags_t dva_flags;
  70         list_t *dva_list;
  71         int dva_error;
  72 } dvma_arg_t;
______unchanged_portion_omitted_
```

```
**********************************************************
   67080 Thu Dec 26 13:50:26 2013
new/usr/src/uts/i86pc/io/mp_platform_common.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/apic_timer.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
```



```
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/cyclic.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/sunndi.h>
#if !defined(__xpv)
#include <sys/hpet.h>
#include <sys/clock.h>
#endif

/*
 *      Local Function Prototypes
 */
static int apic_handle_defconf();
static int apic_parse_mpct(caddr_t mpct, int bypass);
static struct apic_mpfps_hdr *apic_find_fps_sig(caddr_t fptr, int size);
static int apic_checksum(caddr_t bptr, int len);
static int apic_find_bus_type(char *bus);
static int apic_find_bus(int busid);
static struct apic_io_intr *apic_find_io_intr(int irqno);
static int apic_find_free_irq(int start, int end);
struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
static void apic_set_pwroff_method_from_mpcnfhdr(struct apic_mp_cnf_hdr *hdrp);
static void apic_free_apic_cpus(void);
static boolean_t apic_is_ioapic_AMD_813x(uint32_t physaddr);
static int apic_acpi_enter_apicmode(void);

int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
    int child_ipin, struct apic_io_intr **intrp);
int apic_find_bus_id(int bustype);
int apic_find_intin(uchar_t ioapic, uchar_t intin);
void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

int apic_debug_mps_id = 0;      /* 1 - print MPS ID strings */

/* ACPI SCI interrupt configuration; -1 if SCI not used */
int apic_sci_vect = -1;
iflag_t apic_sci_flags;

#if !defined(__xpv)
/* ACPI HPET interrupt configuration; -1 if HPET not used */
int apic_hpet_vect = -1;
iflag_t apic_hpet_flags;
#endif

/*
 * psm name pointer
 */
char *psm_name;

/* ACPI support routines */
static int acpi_probe(char *);
static int apic_acpi_irq_configure(acpi_psm_lnk_t *acpipsmlnkp, dev_info_t *dip,
    int *pci_irqp, iflag_t *intr_flagp);

int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
    int ipin, int *pci_irqp, iflag_t *intr_flagp);
uchar_t acpi_find_ioapic(int irq);
static int acpi_intr_compatible(iflag_t iflag1, iflag_t iflag2);

/* Max wait time (in repetitions) for flags to clear in an RDT entry. */
int apic_max_reps_clear_pending = 1000;
```

```
128 int     apic_intr_policy = INTR_ROUND_ROBIN;

130 int     apic_next_bind_cpu = 1; /* For round robin assignment */
131                                 /* start with cpu 1 */

133 /*
134  * If enabled, the distribution works as follows:
135  * On every interrupt entry, the current ipl for the CPU is set in cpu_info
136  * and the irq corresponding to the ipl is also set in the aci_current array.
137  * interrupt exit and setspl (due to soft interrupts) will cause the current
138  * ipl to be be changed. This is cache friendly as these frequently used
139  * paths write into a per cpu structure.
140  *
141  * Sampling is done by checking the structures for all CPUs and incrementing
142  * the busy field of the irq (if any) executing on each CPU and the busy field
143  * of the corresponding CPU.
144  * In periodic mode this is done on every clock interrupt.
145  * In one-shot mode, this is done thru a cyclic with an interval of
146  * apic_redistribute_sample_interval (default 10 milli sec).
147  *
148  * Every apic_sample_factor_redistribution times we sample, we do computations
149  * to decide which interrupt needs to be migrated (see comments
150  * before apic_intr_redistribute().
151  */

153 /*
154  * Following 3 variables start as % and can be patched or set using an
155  * API to be defined in future. They will be scaled to
156  * sample_factor_redistribution which is in turn set to hertz+1 (in periodic
157  * mode), or 101 in one-shot mode to stagger it away from one sec processing
158  */

160 int     apic_int_busy_mark = 60;
161 int     apic_int_free_mark = 20;
162 int     apic_diff_for_redistribution = 10;

164 /* sampling interval for interrupt redistribution for dynamic migration */
165 int     apic_redistribute_sample_interval = NANOSEC / 100; /* 10 millisec */

167 /*
168  * number of times we sample before deciding to redistribute interrupts
169  * for dynamic migration
170  */
171 int     apic_sample_factor_redistribution = 101;

173 int     apic_redist_cpu_skip = 0;
174 int     apic_num_imbalance = 0;
175 int     apic_num_rebind = 0;

177 /*
178  * Maximum number of APIC CPUs in the system, -1 indicates that dynamic
179  * allocation of CPU ids is disabled.
180  */
181 int     apic_max_nproc = -1;
182 int     apic_nproc = 0;
183 size_t  apic_cpus_size = 0;
184 int     apic_defconf = 0;
185 int     apic_irq_translate = 0;
186 int     apic_spec_rev = 0;
187 int     apic_imcrp = 0;

189 int     apic_use_acpi = 1;      /* 1 = use ACPI, 0 = don't use ACPI */
190 int     apic_use_acpi_madt_only = 0;    /* 1=ONLY use MADT from ACPI */

192 /*
```



```
193  * For interrupt link devices, if apic_unconditional_srs is set, an irq resource
194  * will be assigned (via _SRS). If it is not set, use the current
195  * irq setting (via _CRS), but only if that irq is in the set of possible
196  * irqs (returned by _PRS) for the device.
197  */
198 int     apic_unconditional_srs = 1;

200 /*
201  * For interrupt link devices, if apic_prefer_crs is set when we are
202  * assigning an IRQ resource to a device, prefer the current IRQ setting
203  * over other possible irq settings under same conditions.
204  */

206 int     apic_prefer_crs = 1;

208 uchar_t apic_io_id[MAX_IO_APIC];
209 volatile uint32_t *apicioadr[MAX_IO_APIC];
210 uchar_t apic_io_ver[MAX_IO_APIC];
211 uchar_t apic_io_vectbase[MAX_IO_APIC];
212 uchar_t apic_io_vectend[MAX_IO_APIC];
213 uchar_t apic_reserved_irqlist[MAX_ISA_IRQ + 1];
214 uint32_t apic_physaddr[MAX_IO_APIC];

216 boolean_t ioapic_mask_workaround[MAX_IO_APIC];

218 /*
219  * First available slot to be used as IRQ index into the apic_irq_table
220  * for those interrupts (like MSI/X) that don't have a physical IRQ.
221  */
222 int apic_first_avail_irq  = APIC_FIRST_FREE_IRQ;

224 /*
225  * apic_ioapic_lock protects the ioapics (reg select), the status, temp_bound
226  * and bound elements of cpus_info and the temp_cpu element of irq_struct
227  */
228 lock_t  apic_ioapic_lock;

230 int     apic_io_max = 0;        /* no. of i/o apics enabled */

232 struct apic_io_intr *apic_io_intrp = NULL;
233 static  struct apic_bus *apic_busp;

235 uchar_t apic_resv_vector[MAXIPL+1];

237 char    apic_level_intr[APIC_MAX_VECTOR+1];

239 uint32_t        eisa_level_intr_mask = 0;
240         /* At least MSB will be set if EISA bus */

242 int     apic_pci_bus_total = 0;
243 uchar_t apic_single_pci_busid = 0;

245 /*
246  * airq_mutex protects additions to the apic_irq_table - the first
247  * pointer and any airq_nexts off of that one. It also protects
248  * apic_max_device_irq & apic_min_device_irq. It also guarantees
249  * that share_id is unique as new ids are generated only when new
250  * irq_t structs are linked in. Once linked in the structs are never
251  * deleted. temp_cpu & mps_intr_index field indicate if it is programmed
252  * or allocated. Note that there is a slight gap between allocating in
253  * apic_introp_xlate and programming in addspl.
254  */
255 kmutex_t        airq_mutex;
256 apic_irq_t      *apic_irq_table[APIC_MAX_VECTOR+1];
257 int             apic_max_device_irq = 0;
258 int             apic_min_device_irq = APIC_MAX_VECTOR;
```

```
 260 typedef struct prs_irq_list_ent {
 261         int                     list_prio;
 262         int32_t                 irq;
 263         iflag_t                 intrflags;
 264         acpi_prs_private_t      prsprv;
 265         struct prs_irq_list_ent *next;
 266 } prs_irq_list_t;
______unchanged_portion_omitted_
```

```
**********************************************************
   63988 Thu Dec 26 13:50:26 2013
new/usr/src/uts/i86pc/io/mp_platform_misc.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/inttypes.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/apic_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
```



```
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/cyclic.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/sunndi.h>
#include <sys/hpet.h>
#include <sys/clock.h>

/*
 * Part of mp_platfrom_common.c that's used only by pcplusmp & xpv_psm
 * but not apix.
 * These functions may be moved to xpv_psm later when apix and pcplusmp
 * are merged together
 */

/*
 *      Local Function Prototypes
 */
static void apic_mark_vector(uchar_t oldvector, uchar_t newvector);
static void apic_xlate_vector_free_timeout_handler(void *arg);
static int apic_check_stuck_interrupt(apic_irq_t *irq_ptr, int old_bind_cpu,
    int new_bind_cpu, int apicindex, int intin_no, int which_irq,
    struct ioapic_reprogram_data *drep);
static int apic_setup_irq_table(dev_info_t *dip, int irqno,
    struct apic_io_intr *intrp, struct intrspec *ispec, iflag_t *intr_flagp,
    int type);
static void apic_try_deferred_reprogram(int ipl, int vect);
static void delete_defer_repro_ent(int which_irq);
static void apic_ioapic_wait_pending_clear(int ioapicindex,
    int intin_no);

extern int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
    int ipin, int *pci_irqp, iflag_t *intr_flagp);
extern int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
    int child_ipin, struct apic_io_intr **intrp);
extern uchar_t acpi_find_ioapic(int irq);
extern struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
extern int apic_find_bus_id(int bustype);
extern int apic_find_intin(uchar_t ioapic, uchar_t intin);
extern void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

extern  int apic_sci_vect;
extern  iflag_t apic_sci_flags;
/* ACPI HPET interrupt configuration; -1 if HPET not used */
extern  int apic_hpet_vect;
extern  iflag_t apic_hpet_flags;
extern  int     apic_intr_policy;
extern  char *psm_name;

/*
 * number of bits per byte, from <sys/param.h>
 */
#define UCHAR_MAX       UINT8_MAX

/* Max wait time (in repetitions) for flags to clear in an RDT entry. */
extern int apic_max_reps_clear_pending;

/* The irq # is implicit in the array index: */
struct ioapic_reprogram_data apic_reprogram_info[APIC_MAX_VECTOR+1];
/*
 * APIC_MAX_VECTOR + 1 is the maximum # of IRQs as well. ioapic_reprogram_info
```

```
 127  * is indexed by IRQ number, NOT by vector number.
 128  */

 130 extern  int     apic_int_busy_mark;
 131 extern  int     apic_int_free_mark;
 132 extern  int     apic_diff_for_redistribution;
 133 extern  int     apic_sample_factor_redistribution;
 134 extern  int     apic_redist_cpu_skip;
 135 extern  int     apic_num_imbalance;
 136 extern  int     apic_num_rebind;

 138 /* timeout for xlate_vector, mark_vector */
 139 int     apic_revector_timeout = 16 * 10000; /* 160 millisec */

 141 extern int      apic_defconf;
 142 extern int      apic_irq_translate;

 144 extern int      apic_use_acpi_madt_only;        /* 1=ONLY use MADT from ACPI */

 146 extern  uchar_t apic_io_vectbase[MAX_IO_APIC];

 148 extern  boolean_t ioapic_mask_workaround[MAX_IO_APIC];

 150 /*
 151  * First available slot to be used as IRQ index into the apic_irq_table
 152  * for those interrupts (like MSI/X) that don't have a physical IRQ.
 153  */
 154 extern int apic_first_avail_irq;

 156 /*
 157  * apic_defer_reprogram_lock ensures that only one processor is handling
 158  * deferred interrupt programming at *_intr_exit time.
 159  */
 160 static  lock_t  apic_defer_reprogram_lock;

 162 /*
 163  * The current number of deferred reprogrammings outstanding
 164  */
 165 uint_t  apic_reprogram_outstanding = 0;

 167 #ifdef DEBUG
 168 /*
 169  * Counters that keep track of deferred reprogramming stats
 170  */
 171 uint_t  apic_intr_deferrals = 0;
 172 uint_t  apic_intr_deliver_timeouts = 0;
 173 uint_t  apic_last_ditch_reprogram_failures = 0;
 174 uint_t  apic_deferred_setup_failures = 0;
 175 uint_t  apic_defer_repro_total_retries = 0;
 176 uint_t  apic_defer_repro_successes = 0;
 177 uint_t  apic_deferred_spurious_enters = 0;
 178 #endif

 180 extern  int     apic_io_max;
 181 extern  struct apic_io_intr *apic_io_intrp;

 183 uchar_t apic_vector_to_irq[APIC_MAX_VECTOR+1];

 185 extern  uint32_t        eisa_level_intr_mask;
 186         /* At least MSB will be set if EISA bus */

 188 extern  int     apic_pci_bus_total;
 189 extern  uchar_t apic_single_pci_busid;

 191 /*
 192  * Following declarations are for revectoring; used when ISRs at different
```



```
 193  * IPLs share an irq.
 194  */
 195 static  lock_t  apic_revector_lock;
 196 int     apic_revector_pending = 0;
 197 static  uchar_t *apic_oldvec_to_newvec;
 198 static  uchar_t *apic_newvec_to_oldvec;

 200 /* ACPI Interrupt Source Override Structure ptr */
 201 extern ACPI_MADT_INTERRUPT_OVERRIDE *acpi_isop;
 202 extern int acpi_iso_cnt;

 204 /*
 205  * Auto-configuration routines
 206  */

 208 /*
 209  * Initialise vector->ipl and ipl->pri arrays. level_intr and irqtable
 210  * are also set to NULL. vector->irq is set to a value which cannot map
 211  * to a real irq to show that it is free.
 212  */
 213 void
 214 apic_init_common(void)
 215 {
 216         int     i, j, indx;
 217         int     *iptr;

 219         /*
 220          * Initialize apic_ipls from apic_vectortoipl.  This array is
 221          * used in apic_intr_enter to determine the IPL to use for the
 222          * corresponding vector.  On some systems, due to hardware errata
 223          * and interrupt sharing, the IPL may not correspond to the IPL listed
 224          * in apic_vectortoipl (see apic_addspl and apic_delspl).
 225          */
 226         for (i = 0; i < (APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL); i++) {
 227                 indx = i * APIC_VECTOR_PER_IPL;

 229                 for (j = 0; j < APIC_VECTOR_PER_IPL; j++, indx++)
 230                         apic_ipls[indx] = apic_vectortoipl[i];
 231         }

 233         /* cpu 0 is always up (for now) */
 234         apic_cpus[0].aci_status = APIC_CPU_ONLINE | APIC_CPU_INTR_ENABLE;

 236         iptr = (int *)&apic_irq_table[0];
 237         for (i = 0; i <= APIC_MAX_VECTOR; i++) {
 238                 apic_level_intr[i] = 0;
 239                 *iptr++ = NULL;
 240                 apic_vector_to_irq[i] = APIC_RESV_IRQ;

 242                 /* These *must* be initted to B_TRUE! */
 243                 apic_reprogram_info[i].done = B_TRUE;
 244                 apic_reprogram_info[i].irqp = NULL;
 245                 apic_reprogram_info[i].tries = 0;
 246                 apic_reprogram_info[i].bindcpu = 0;
 247         }

 249         /*
 250          * Allocate a dummy irq table entry for the reserved entry.
 251          * This takes care of the race between removing an irq and
 252          * clock detecting a CPU in that irq during interrupt load
 253          * sampling.
 254          */
 255         apic_irq_table[APIC_RESV_IRQ] =
 256             kmem_zalloc(sizeof (apic_irq_t), KM_SLEEP);

 258         mutex_init(&airq_mutex, NULL, MUTEX_DEFAULT, NULL);
```

```
 259 }
______unchanged_portion_omitted_
```

```
**********************************************************
   11718 Thu Dec 26 13:50:27 2013
new/usr/src/uts/i86pc/io/pciex/npe_misc.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2013 PALO, Richard. All rights reserved.
 */

/*
 *      Library file that has miscellaneous support for npe(7d)
 */

#include <sys/conf.h>
#include <sys/pci.h>
#include <sys/sunndi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpi/acpi_pci.h>
#include <sys/acpica.h>
#include <sys/pci_cap.h>
#include <sys/pcie_impl.h>
#include <sys/x86_archext.h>
#include <io/pciex/pcie_nvidia.h>
#include <io/pciex/pcie_nb5000.h>
#include <sys/pci_cfgacc_x86.h>
#include <sys/cpuvar.h>

/*
 * Prototype declaration
 */
void    npe_query_acpi_mcfg(dev_info_t *dip);
void    npe_ck804_fix_aer_ptr(ddi_acc_handle_t cfg_hdl);
int     npe_disable_empty_bridges_workaround(dev_info_t *child);
void    npe_nvidia_error_workaround(ddi_acc_handle_t cfg_hdl);
void    npe_intel_error_workaround(ddi_acc_handle_t cfg_hdl);
boolean_t npe_is_child_pci(dev_info_t *dip);
int     npe_enable_htmsi(ddi_acc_handle_t cfg_hdl);
void    npe_enable_htmsi_children(dev_info_t *dip);

int     npe_enable_htmsi_flag = 1;

/*
```



```
 * Default ecfga base address
 */
int64_t npe_default_ecfga_base = 0xE0000000;

extern uint32_t npe_aer_uce_mask;

/*
 * Query the MCFG table using ACPI.  If MCFG is found, setup the
 * 'ecfg' property accordingly.  Otherwise, set the values
 * to the default values.
 */
void
npe_query_acpi_mcfg(dev_info_t *dip)
{
        ACPI_TABLE_HEADER *mcfgp;
        ACPI_MCFG_ALLOCATION *cfg_baap;
        MCFG_TABLE *mcfgp;
        CFG_BASE_ADDR_ALLOC *cfg_baap;
        char *cfg_baa_endp;
        int64_t ecfginfo[4];
        int ecfg_found = 0;

        /* Query the MCFG table using ACPI */
        if (AcpiGetTable(ACPI_SIG_MCFG, 1, &mcfgp) == AE_OK) {
        if (AcpiGetTable(ACPI_SIG_MCFG, 1,
            (ACPI_TABLE_HEADER **)&mcfgp) == AE_OK) {

            cfg_baap = (ACPI_MCFG_ALLOCATION *)((ACPI_TABLE_MCFG *)mcfgp + 1);
                cfg_baap = (CFG_BASE_ADDR_ALLOC *)mcfgp->CfgBaseAddrAllocList;
            cfg_baa_endp = ((char *)mcfgp) + mcfgp->Length;

                while ((char *)cfg_baap < cfg_baa_endp) {
                        if (cfg_baap->Address != (uint64_t)0 &&
                            cfg_baap->PciSegment == 0) {
                        if (cfg_baap->base_addr != (uint64_t)0 &&
                            cfg_baap->segment == 0) {
                                /*
                                 * Set up the 'ecfg' property to hold
                                 * base_addr, segment, and first/last bus.
                                 * We only do the first entry that maps
                                 * segment 0; nonzero segments are not yet
                                 * known, or handled.  If they appear,
                                 * we'll need to figure out which bus node
                                 * should have which entry by examining the
                                 * ACPI _SEG method on each bus node.
                                 */
                                ecfginfo[0] = cfg_baap->Address;
                                ecfginfo[1] = cfg_baap->PciSegment;
                                ecfginfo[2] = cfg_baap->StartBusNumber;
                                ecfginfo[3] = cfg_baap->EndBusNumber;
                                ecfginfo[0] = cfg_baap->base_addr;
                                ecfginfo[1] = cfg_baap->segment;
                                ecfginfo[2] = cfg_baap->start_bno;
                                ecfginfo[3] = cfg_baap->end_bno;
                                (void) ndi_prop_update_int64_array(
                                    DDI_DEV_T_NONE, dip, "ecfg",
                                    ecfginfo, 4);
                                ecfg_found = 1;
                                break;
                        }
                        cfg_baap++;
                }
        }
        if (ecfg_found)
                return;
        /*
```

```
114          * If MCFG is not found or ecfga_base is not found in MCFG table,
115          * set the property to the default values.
116          */
117         ecfginfo[0] = npe_default_ecfga_base;
118         ecfginfo[1] = 0;                /* segment 0 */
119         ecfginfo[2] = 0;                /* first bus 0 */
120         ecfginfo[3] = 0xff;             /* last bus ff */
121         (void) ndi_prop_update_int64_array(DDI_DEV_T_NONE, dip,
122             "ecfg", ecfginfo, 4);
123 }
```

unchanged_portion_omitted_

```
**********************************************************
   36120 Thu Dec 26 13:50:27 2013
new/usr/src/uts/i86pc/io/pcplusmp/apic.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright (c) 2013, Joyent, Inc.  All rights reserved.
 */

/*
 * To understand how the pcplusmp module interacts with the interrupt subsystem
 * read the theory statement in uts/i86pc/os/intr.c.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
```



```
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/cyclic.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/hpet.h>
#include <sys/apic_common.h>
#include <sys/apic_timer.h>

/*
 *      Local Function Prototypes
 */
static void apic_init_intr(void);

/*
 *      standard MP entries
 */
static int      apic_probe(void);
static int      apic_getclkirq(int ipl);
static void     apic_init(void);
static void     apic_picinit(void);
static int      apic_post_cpu_start(void);
static int      apic_intr_enter(int ipl, int *vect);
static void     apic_setspl(int ipl);
static void     x2apic_setspl(int ipl);
static int      apic_addspl(int ipl, int vector, int min_ipl, int max_ipl);
static int      apic_delspl(int ipl, int vector, int min_ipl, int max_ipl);
static int      apic_disable_intr(processorid_t cpun);
static void     apic_enable_intr(processorid_t cpun);
static int              apic_get_ipivect(int ipl, int type);
static void     apic_post_cyclic_setup(void *arg);

/*
 * The following vector assignments influence the value of ipltopri and
 * vectortoipl. Note that vectors 0 - 0x1f are not used. We can program
 * idle to 0 and IPL 0 to 0xf to differentiate idle in case
 * we care to do so in future. Note some IPLs which are rarely used
 * will share the vector ranges and heavily used IPLs (5 and 6) have
 * a wide range.
 *
 * This array is used to initialize apic_ipls[] (in apic_init()).
 *
 *      IPL             Vector range.           as passed to intr_enter
 *      0               none.
 *      1,2,3           0x20-0x2f               0x0-0xf
 *      4               0x30-0x3f               0x10-0x1f
 *      5               0x40-0x5f               0x20-0x3f
 *      6               0x60-0x7f               0x40-0x5f
```

```
 127  *      7,8,9           0x80-0x8f               0x60-0x6f
 128  *      10              0x90-0x9f               0x70-0x7f
 129  *      11              0xa0-0xaf               0x80-0x8f
 130  *      ...             ...
 131  *      15              0xe0-0xef               0xc0-0xcf
 132  *      15              0xf0-0xff               0xd0-0xdf
 133  */
 134 uchar_t apic_vectortoipl[APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL] = {
 135         3, 4, 5, 5, 6, 6, 9, 10, 11, 12, 13, 14, 15, 15
 136 };
_____unchanged_portion_omitted_
```

```
**********************************************************
   43963 Thu Dec 26 13:50:28 2013
new/usr/src/uts/i86pc/io/pcplusmp/apic_common.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2013, Joyent, Inc.  All rights reserved.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
```



```
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/hpet.h>
#include <sys/apic_common.h>
#include <sys/apic_timer.h>

static void     apic_record_ioapic_rdt(void *intrmap_private,
                    ioapic_rdt_t *irdt);
static void     apic_record_msi(void *intrmap_private, msi_regs_t *mregs);

/*
 * Common routines between pcplusmp & apix (taken from apic.c).
 */

int     apic_clkinit(int);
hrtime_t apic_gethrtime(void);
void    apic_send_ipi(int, int);
void    apic_set_idlecpu(processorid_t);
void    apic_unset_idlecpu(processorid_t);
void    apic_shutdown(int, int);
void    apic_preshutdown(int, int);
processorid_t   apic_get_next_processorid(processorid_t);

hrtime_t apic_gettime();

enum apic_ioapic_method_type apix_mul_ioapic_method = APIC_MUL_IOAPIC_PCPLUSMP;

/* Now the ones for Dynamic Interrupt distribution */
int     apic_enable_dynamic_migration = 0;

/* maximum loop count when sending Start IPIs. */
int apic_sipi_max_loop_count = 0x1000;

/*
 * These variables are frequently accessed in apic_intr_enter(),
 * apic_intr_exit and apic_setspl, so group them together
 */
volatile uint32_t *apicadr =  NULL;     /* virtual addr of local APIC   */
int apic_setspl_delay = 1;              /* apic_setspl - delay enable   */
int apic_clkvect;

/* vector at which error interrupts come in */
int apic_errvect;
int apic_enable_error_intr = 1;
int apic_error_display_delay = 100;

/* vector at which performance counter overflow interrupts come in */
int apic_cpcovf_vect;
int apic_enable_cpcovf_intr = 1;

/* vector at which CMCI interrupts come in */
int apic_cmci_vect;
extern int cmi_enable_cmci;
extern void cmi_cmci_trap(void);
```

```
 127 kmutex_t cmci_cpu_setup_lock;   /* protects cmci_cpu_setup_registered */
 128 int cmci_cpu_setup_registered;

 130 /* number of CPUs in power-on transition state */
 131 static int apic_poweron_cnt = 0;
 132 lock_t apic_mode_switch_lock;

 134 /*
 135  * Patchable global variables.
 136  */
 137 int     apic_forceload = 0;

 139 int     apic_coarse_hrtime = 1;         /* 0 - use accurate slow gethrtime() */

 141 int     apic_flat_model = 0;            /* 0 - clustered. 1 - flat */
 142 int     apic_panic_on_nmi = 0;
 143 int     apic_panic_on_apic_error = 0;

 145 int     apic_verbose = 0;       /* 0x1ff */

 147 #ifdef DEBUG
 148 int     apic_debug = 0;
 149 int     apic_restrict_vector = 0;

 151 int     apic_debug_msgbuf[APIC_DEBUG_MSGBUFSIZE];
 152 int     apic_debug_msgbufindex = 0;

 154 #endif /* DEBUG */

 156 uint_t apic_nticks = 0;
 157 uint_t apic_skipped_redistribute = 0;

 159 uint_t last_count_read = 0;
 160 lock_t  apic_gethrtime_lock;
 161 volatile int    apic_hrtime_stamp = 0;
 162 volatile hrtime_t apic_nsec_since_boot = 0;

 164 static  hrtime_t        apic_last_hrtime = 0;
 165 int             apic_hrtime_error = 0;
 166 int             apic_remote_hrterr = 0;
 167 int             apic_num_nmis = 0;
 168 int             apic_apic_error = 0;
 169 int             apic_num_apic_errors = 0;
 170 int             apic_num_cksum_errors = 0;

 172 int     apic_error = 0;

 174 static  int     apic_cmos_ssb_set = 0;

 176 /* use to make sure only one cpu handles the nmi */
 177 lock_t  apic_nmi_lock;
 178 /* use to make sure only one cpu handles the error interrupt */
 179 lock_t  apic_error_lock;

 181 static  struct {
 182         uchar_t cntl;
 183         uchar_t data;
 184 } aspen_bmc[] = {
_____unchanged_portion_omitted_
```

```
**********************************************************
    9947 Thu Dec 26 13:50:28 2013
new/usr/src/uts/i86pc/io/ppm/acpippm.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2011 Bayard G. Bell. All rights reserved.
 */

#include <sys/types.h>
#include <sys/conf.h>
#include <sys/open.h>
#include <sys/modctl.h>
#include <sys/promif.h>
#include <sys/stat.h>
#include <sys/ddi_impldefs.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/epm.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_types.h>

/*
 *      ACPI Power Management Driver
 *
 *      acpippm deals with those bits of ppm functionality that
 *      must be mediated by ACPI
 *
 *      The routines in this driver is referenced by Platform
 *      Power Management driver of X86 workstation systems.
 *      acpippm driver is loaded because it is listed as a platform driver
 *      It is initially configured as a pseudo driver.
 */
extern void pc_tod_set_rtc_offsets(ACPI_TABLE_FADT *);
extern int acpica_use_safe_delay;

/*
 * Configuration Function prototypes and data structures
 */
static int      appm_attach(dev_info_t *, ddi_attach_cmd_t);
static int      appm_detach(dev_info_t *, ddi_detach_cmd_t);
```



```
static int      appm_getinfo(dev_info_t *, ddi_info_cmd_t, void *, void **);
static int      appm_open(dev_t *dev_p, int flag, int otyp, cred_t *cred_p);
static int      appm_close(dev_t dev, int flag, int otyp, cred_t *cred_p);
static int      appm_ioctl(dev_t, int, intptr_t, int, cred_t *, int *);

/*
 * Configuration data structures
 */
static struct cb_ops appm_cbops = {
        appm_open,              /* open */
        appm_close,             /* close */
        nodev,                  /* strategy */
        nodev,                  /* print */
        nodev,                  /* dump */
        nodev,                  /* read */
        nodev,                  /* write */
        appm_ioctl,             /* ioctl */
        nodev,                  /* devmap */
        nodev,                  /* mmap */
        nodev,                  /* segmap */
        nochpoll,               /* chpoll */
        ddi_prop_op,            /* prop_op */
        NULL,                   /* stream */
        D_MP | D_NEW,           /* flag */
        CB_REV,                 /* rev */
        nodev,                  /* aread */
        nodev,                  /* awrite */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
    4338 Thu Dec 26 13:50:29 2013
new/usr/src/uts/i86pc/io/ppm/acpisleep.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#include <sys/types.h>
#include <sys/smp_impldefs.h>
#include <sys/promif.h>

#include <sys/kmem.h>
#include <sys/archsystm.h>
#include <sys/cpuvar.h>
#include <sys/pte.h>
#include <vm/seg_kmem.h>
#include <sys/epm.h>
#include <sys/cpr.h>
#include <sys/machsystm.h>
#include <sys/clock.h>

#include <sys/cpr_wakecode.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>

#ifdef OLDPMCODE
#include "acpi.h"
#endif

#include        <sys/x86_archext.h>
#include        <sys/reboot.h>
#include        <sys/cpu_module.h>
#include        <sys/kdi.h>

/*
 * S3 stuff
 */

int acpi_rtc_wake = 0x0;                /* wake in N seconds */

#if 0   /* debug */
static uint8_t  branchbuf[64 * 1024];   /* for the HDT branch trace stuff */
```



```
#endif  /* debug */

extern int boothowto;

#define BOOTCPU 0       /* cpu 0 is always the boot cpu */

extern void             kernel_wc_code(void);
extern tod_ops_t        *tod_ops;
extern int flushes_require_xcalls;
extern int tsc_gethrtime_enable;

extern cpuset_t cpu_ready_set;
extern void *(*cpu_pause_func)(void *);



/*
 * This is what we've all been waiting for!
 */
int
acpi_enter_sleepstate(s3a_t *s3ap)
{
        ACPI_PHYSICAL_ADDRESS   wakephys = s3ap->s3a_wakephys;
        caddr_t                 wakevirt = rm_platter_va;
        /*LINTED*/
        wakecode_t              *wp = (wakecode_t *)wakevirt;
        uint_t                  Sx = s3ap->s3a_state;

        PT(PT_SWV);
        /* Set waking vector */
        if (AcpiSetFirmwareWakingVector(wakephys) != AE_OK) {
                PT(PT_SWV_FAIL);
                PMD(PMD_SX, ("Can't SetFirmwareWakingVector(%lx)\n",
                    (long)wakephys))
                goto insomnia;
        }

        PT(PT_EWE);
        /* Enable wake events */
        if (AcpiEnableEvent(ACPI_EVENT_POWER_BUTTON, 0) != AE_OK) {
                PT(PT_EWE_FAIL);
                PMD(PMD_SX, ("Can't EnableEvent(POWER_BUTTON)\n"))
        }
        if (acpi_rtc_wake > 0) {
                /* clear the RTC bit first */
                (void) AcpiWriteBitRegister(ACPI_BITREG_RT_CLOCK_STATUS, 1);
                PT(PT_RTCW);
                if (AcpiEnableEvent(ACPI_EVENT_RTC, 0) != AE_OK) {
                        PT(PT_RTCW_FAIL);
                        PMD(PMD_SX, ("Can't EnableEvent(RTC)\n"))
                }

                /*
                 * Set RTC to wake us in a wee while.
                 */
                mutex_enter(&tod_lock);
                PT(PT_TOD);
                TODOP_SETWAKE(tod_ops, acpi_rtc_wake);
                mutex_exit(&tod_lock);
        }

        /*
         * Prepare for sleep ... could've done this earlier?
         */
        PT(PT_SXP);
        PMD(PMD_SX, ("Calling AcpiEnterSleepStatePrep(%d) ...\n", Sx))
```

```
127         if (AcpiEnterSleepStatePrep(Sx) != AE_OK) {
128                 PMD(PMD_SX, ("... failed\n!"))
129                 goto insomnia;
130         }

132         switch (s3ap->s3a_test_point) {
133         case DEVICE_SUSPEND_TO_RAM:
134         case FORCE_SUSPEND_TO_RAM:
135         case LOOP_BACK_PASS:
136                 return (0);
137         case LOOP_BACK_FAIL:
138                 return (1);
139         default:
140                 ASSERT(s3ap->s3a_test_point == LOOP_BACK_NONE);
141         }

143         /*
144          * Tell the hardware to sleep.
145          */
146         PT(PT_SXE);
147         PMD(PMD_SX, ("Calling AcpiEnterSleepState(%d) ...\n", Sx))
148         if (AcpiEnterSleepState(Sx) != AE_OK) {
149                 PT(PT_SXE_FAIL);
150                 PMD(PMD_SX, ("... failed!\n"))
151         }

153 insomnia:
154         PT(PT_INSOM);
155         /* cleanup is done in the caller */
156         return (1);
157 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   27091 Thu Dec 26 13:50:29 2013
new/usr/src/uts/i86pc/io/psm/psm_common.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2013 PALO, Richard. All rights reserved.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/cmn_err.h>
#include <sys/promif.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/sunddi.h>
#include <sys/ddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/debug.h>
#include <sys/psm_common.h>
#include <sys/sunndi.h>
#include <sys/ksynch.h>

/* Global configurables */

char *psm_module_name;  /* used to store name of psm module */

/*
 * acpi_irq_check_elcr: when set elcr will also be consulted for building
 * the reserved irq list.  When 0 (false), the existing state of the ELCR
 * is ignored when selecting a vector during IRQ translation, and the ELCR
 * is programmed to the proper setting for the type of bus (level-triggered
 * for PCI, edge-triggered for non-PCI).  When non-zero (true), vectors
 * set to edge-mode will not be used when in PIC-mode.  The default value
 * is 0 (false).  Note that ACPI's SCI vector is always set to conform to
 * ACPI-specification regardless of this.
 *
 */
int acpi_irq_check_elcr = 0;

int psm_verbose = 0;
```



```
#define PSM_VERBOSE_IRQ(fmt)    \
                if (psm_verbose & PSM_VERBOSE_IRQ_FLAG) \
                        cmn_err fmt;

#define PSM_VERBOSE_POWEROFF(fmt)  \
                if (psm_verbose & PSM_VERBOSE_POWEROFF_FLAG || \
                    psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) \
                        prom_printf fmt;

#define PSM_VERBOSE_POWEROFF_PAUSE(fmt) \
                if (psm_verbose & PSM_VERBOSE_POWEROFF_FLAG || \
                    psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) {\
                        prom_printf fmt; \
                        if (psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) \
                                (void) goany(); \
                }


/* Local storage */
static ACPI_HANDLE acpi_sbobj = NULL;
static kmutex_t acpi_irq_cache_mutex;

/*
 * irq_cache_table is a list that serves a two-key cache. It is used
 * as a pci busid/devid/ipin <-> irq cache and also as a acpi
 * interrupt lnk <-> irq cache.
 */
static irq_cache_t *irq_cache_table;

#define IRQ_CACHE_INITLEN       20
static int irq_cache_len = 0;
static int irq_cache_valid = 0;

static int acpi_get_gsiv(dev_info_t *dip, ACPI_HANDLE pciobj, int devno,
        int ipin, int *pci_irqp, iflag_t *iflagp,  acpi_psm_lnk_t *acpipsmlnkp);

static int acpi_eval_lnk(dev_info_t *dip, char *lnkname,
    int *pci_irqp, iflag_t *intr_flagp, acpi_psm_lnk_t *acpipsmlnkp);

static int acpi_get_irq_lnk_cache_ent(ACPI_HANDLE lnkobj, int *pci_irqp,
    iflag_t *intr_flagp);

extern int goany(void);


#define NEXT_PRT_ITEM(p)        \
                (void *)(((char *)(p)) + (p)->Length)

static int
acpi_get_gsiv(dev_info_t *dip, ACPI_HANDLE pciobj, int devno, int ipin,
    int *pci_irqp, iflag_t *intr_flagp, acpi_psm_lnk_t *acpipsmlnkp)
{
        ACPI_BUFFER rb;
        ACPI_PCI_ROUTING_TABLE *prtp;
        int status;
        int dev_adr;

        /*
         * Get the IRQ routing table
         */
        rb.Pointer = NULL;
        rb.Length = ACPI_ALLOCATE_BUFFER;
        if (AcpiGetIrqRoutingTable(pciobj, &rb) != AE_OK) {
                return (ACPI_PSM_FAILURE);
        }
```

```
126         status = ACPI_PSM_FAILURE;
127         dev_adr = (devno << 16 | 0xffff);
128         for (prtp = rb.Pointer; prtp->Length != 0; prtp = NEXT_PRT_ITEM(prtp)) {
129                 /* look until a matching dev/pin is found */
130                 if (dev_adr != prtp->Address || ipin != prtp->Pin)
131                         continue;

133                 /* NULL Source name means index is GSIV */
134                 if (*prtp->Source == 0) {
135                         intr_flagp->intr_el = INTR_EL_LEVEL;
136                         intr_flagp->intr_po = INTR_PO_ACTIVE_LOW;
137                         ASSERT(pci_irqp != NULL);
138                         *pci_irqp = prtp->SourceIndex;
139                         status = ACPI_PSM_SUCCESS;
140                 } else
141                         status = acpi_eval_lnk(dip, prtp->Source, pci_irqp,
142                             intr_flagp, acpipsmlnkp);

144                 break;

146         }

148         AcpiOsFree(rb.Pointer);
149         return (status);
150 }
```

unchanged_portion_omitted_

```
449 /*
450  * Sets the irq resource of the lnk object to the requested irq value.
451  *
452  * Returns ACPI_PSM_SUCCESS on success, ACPI_PSM_FAILURE upon failure.
453  */
454 int
455 acpi_set_irq_resource(acpi_psm_lnk_t *acpipsmlnkp, int irq)
456 {
457         ACPI_BUFFER     rsb;
458         ACPI_RESOURCE   *resp;
459         ACPI_RESOURCE   *srsp;
460         ACPI_HANDLE lnkobj;
461         int srs_len, status;

463         ASSERT(acpipsmlnkp != NULL);

465         lnkobj = acpipsmlnkp->lnkobj;

467         /*
468          * Fetch the possible resources for the link
469          */

471         rsb.Pointer = NULL;
472         rsb.Length = ACPI_ALLOCATE_BUFFER;
473         status = AcpiGetPossibleResources(lnkobj, &rsb);
474         if (status != AE_OK) {
475                 cmn_err(CE_WARN, "!psm: set_irq: _PRS failed");
476                 return (ACPI_PSM_FAILURE);
477         }

479         /*
480          * Find an IRQ resource descriptor to use as template
481          */
482         srsp = NULL;
483         for (resp = rsb.Pointer; resp->Type != ACPI_RESOURCE_TYPE_END_TAG;
484             resp = ACPI_NEXT_RESOURCE(resp)) {
485                 if ((resp->Type == ACPI_RESOURCE_TYPE_IRQ) ||
486                     (resp->Type == ACPI_RESOURCE_TYPE_EXTENDED_IRQ)) {
487                         ACPI_RESOURCE *endtag;
```



```
488                         /*
489                          * Allocate enough room for this resource entry
490                          * and one end tag following it
491                          */
492                         srs_len = resp->Length + ACPI_RS_SIZE_NO_DATA;
491                         srs_len = resp->Length + sizeof (*endtag);
493                         srsp = kmem_zalloc(srs_len, KM_SLEEP);
494                         bcopy(resp, srsp, resp->Length);
495                         endtag = ACPI_NEXT_RESOURCE(srsp);
496                         endtag->Type = ACPI_RESOURCE_TYPE_END_TAG;
497                         endtag->Length = ACPI_RS_SIZE_NO_DATA;
496                         endtag->Length = 0;
498                         break;  /* drop out of the loop */
499                 }
500         }

502         /*
503          * We're done with the PRS values, toss 'em lest we forget
504          */
505         AcpiOsFree(rsb.Pointer);

507         if (srsp == NULL)
508                 return (ACPI_PSM_FAILURE);

510         /*
511          * The Interrupts[] array is always at least one entry
512          * long; see the definition of ACPI_RESOURCE.
513          */
514         switch (srsp->Type) {
515         case ACPI_RESOURCE_TYPE_IRQ:
516                 srsp->Data.Irq.InterruptCount = 1;
517                 srsp->Data.Irq.Interrupts[0] = (uint8_t)irq;
518                 break;
519         case ACPI_RESOURCE_TYPE_EXTENDED_IRQ:
520                 srsp->Data.ExtendedIrq.InterruptCount = 1;
521                 srsp->Data.ExtendedIrq.Interrupts[0] = irq;
522                 break;
523         }

525         rsb.Pointer = srsp;
526         rsb.Length = srs_len;
527         status = AcpiSetCurrentResources(lnkobj, &rsb);
528         kmem_free(srsp, srs_len);
529         if (status != AE_OK) {
530                 cmn_err(CE_WARN, "!psm: set_irq: _SRS failed");
531                 return (ACPI_PSM_FAILURE);
532         }

534         if (acpica_eval_int(lnkobj, "_STA", &status) == AE_OK) {
535                 acpipsmlnkp->device_status = (uchar_t)status;
536                 return (ACPI_PSM_SUCCESS);
537         } else
538                 return (ACPI_PSM_FAILURE);
539 }
```

unchanged_portion_omitted_

```
**********************************************************
   11730 Thu Dec 26 13:50:30 2013
new/usr/src/uts/i86pc/io/todpc_subr.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2012 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1990, 1991 UNIX System Laboratories, Inc. */
/*      Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T   */
/*        All Rights Reserved   */

/*      Copyright (c) 1987, 1988 Microsoft Corporation  */
/*        All Rights Reserved   */

#include <sys/param.h>
#include <sys/time.h>
#include <sys/systm.h>

#include <sys/cpuvar.h>
#include <sys/clock.h>
#include <sys/debug.h>
#include <sys/rtc.h>
#include <sys/archsystm.h>
#include <sys/sysmacros.h>
#include <sys/lockstat.h>
#include <sys/stat.h>
#include <sys/sunddi.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

static int todpc_rtcget(unsigned char *buf);
static void todpc_rtcput(unsigned char *buf);

#define CLOCK_RES       1000            /* 1 microsec in nanosecs */

int clock_res = CLOCK_RES;

/*
```



```
 * The minimum sleep time till an alarm can be fired.
 * This can be tuned in /etc/system, but if the value is too small,
 * there is a danger that it will be missed if it takes too long to
 * get from the set point to sleep.  Or that it can fire quickly, and
 * generate a power spike on the hardware.  And small values are
 * probably only usefull for test setups.
 */
int clock_min_alarm = 4;

/*
 * Machine-dependent clock routines.
 */

extern long gmt_lag;

struct rtc_offset {
        int8_t  loaded;
        uint8_t day_alrm;
        uint8_t mon_alrm;
        uint8_t century;
};
______unchanged_portion_omitted_
```

```
**********************************************************
   17867 Thu Dec 26 13:50:30 2013
new/usr/src/uts/i86pc/io/tzmon/tzmon.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Solaris x86 ACPI ThermalZone Monitor
 */


#include <sys/errno.h>
#include <sys/conf.h>
#include <sys/modctl.h>
#include <sys/open.h>
#include <sys/stat.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ksynch.h>
#include <sys/uadmin.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/sdt.h>

#include "tzmon.h"


#define TZMON_ENUM_TRIP_POINTS  1
#define TZMON_ENUM_DEV_LISTS    2
#define TZMON_ENUM_ALL          (TZMON_ENUM_TRIP_POINTS | TZMON_ENUM_DEV_LISTS)

/*
 * TZ_TASKQ_NAME_LEN is precisely the length of the string "AcpiThermalMonitor"
 * plus a two-digit instance number plus a NULL.  If the taskq name is changed
 * (particularly if it is lengthened), then this value needs to change.
 */
#define TZ_TASKQ_NAME_LEN       21

/*
 * Kelvin to Celsius conversion
```



```
 * The formula for converting degrees Kelvin to degrees Celsius is
 * C = K - 273.15 (we round to 273.2).  The unit for thermal zone
 * temperatures is tenths of a degree Kelvin.  Use tenth of a degree
 * to convert, then make a whole number out of it.
 */
#define K_TO_C(temp)            (((temp) - 2732) / 10)


/* cb_ops or dev_ops forward declarations */
static  int     tzmon_getinfo(dev_info_t *dip, ddi_info_cmd_t infocmd,
    void *arg, void **result);
static  int     tzmon_attach(dev_info_t *dip, ddi_attach_cmd_t cmd);
static  int     tzmon_detach(dev_info_t *dip, ddi_detach_cmd_t cmd);

/* other forward declarations */
static void tzmon_notify_zone(ACPI_HANDLE obj, UINT32 val, void *ctx);
static void tzmon_eval_int(ACPI_HANDLE obj, char *method, int *rv);
static thermal_zone_t *tzmon_alloc_zone();
static void tzmon_free_zone_list();
static void tzmon_discard_buffers(thermal_zone_t *tzp);
static void tzmon_enumerate_zone(ACPI_HANDLE obj, thermal_zone_t *tzp,
        int enum_flag);
static ACPI_STATUS tzmon_zone_callback(ACPI_HANDLE obj, UINT32 nest,
    void *ctx, void **rv);
static void tzmon_find_zones(void);
static void tzmon_monitor(void *ctx);
static void tzmon_set_power_device(ACPI_HANDLE dev, int on_off, char *tz_name);
static void tzmon_set_power(ACPI_BUFFER devlist, int on_off, char *tz_name);
static void tzmon_eval_zone(thermal_zone_t *tzp);
static void tzmon_do_shutdown(void);

extern void halt(char *);

static struct cb_ops    tzmon_cb_ops = {
        nodev,                  /* no open routine      */
        nodev,                  /* no close routine     */
        nodev,                  /* not a block driver   */
        nodev,                  /* no print routine     */
        nodev,                  /* no dump routine      */
        nodev,                  /* no read routine      */
        nodev,                  /* no write routine     */
        nodev,                  /* no ioctl routine     */
        nodev,                  /* no devmap routine    */
        nodev,                  /* no mmap routine      */
        nodev,                  /* no segmap routine    */
        nochpoll,               /* no chpoll routine    */
        ddi_prop_op,
        0,                      /* not a STREAMS driver */
        D_NEW | D_MP,           /* safe for multi-thread/multi-processor */
};
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    1456 Thu Dec 26 13:50:30 2013
new/usr/src/uts/i86pc/os/acpi_stubs.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/types.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

/*
 * This file contains ACPI functions that are needed by the kernel before
 * the ACPI module is loaded.  Any functions or definitions need to be
 * able to deal with the possibility that ACPI doesn't get loaded, or
 * doesn't contain the required method.
 */

int (*acpi_fp_setwake)();

/*
 *
 */
int
acpi_ddi_setwake(dev_info_t *dip, int level)
{
        if (acpi_fp_setwake == NULL)
                return (AE_ERROR);

        return ((*acpi_fp_setwake)(dip, level));
}
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   27105 Thu Dec 26 13:50:31 2013
new/usr/src/uts/i86pc/os/cpr_impl.c
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
  23  */

  25 /*
  26  * Platform specific implementation code
  27  * Currently only suspend to RAM is supported (ACPI S3)
  28  */

  30 #define SUNDDI_IMPL

  32 #include <sys/types.h>
  33 #include <sys/promif.h>
  34 #include <sys/prom_isa.h>
  35 #include <sys/prom_plat.h>
  36 #include <sys/cpuvar.h>
  37 #include <sys/pte.h>
  38 #include <vm/hat.h>
  39 #include <vm/page.h>
  40 #include <vm/as.h>
  41 #include <sys/cpr.h>
  42 #include <sys/kmem.h>
  43 #include <sys/clock.h>
  44 #include <sys/kmem.h>
  45 #include <sys/panic.h>
  46 #include <vm/seg_kmem.h>
  47 #include <sys/cpu_module.h>
  48 #include <sys/callb.h>
  49 #include <sys/machsystm.h>
  50 #include <sys/vmsystm.h>
  51 #include <sys/systm.h>
  52 #include <sys/archsystm.h>
  53 #include <sys/stack.h>
  54 #include <sys/fs/ufs_fs.h>
  55 #include <sys/memlist.h>
  56 #include <sys/bootconf.h>
  57 #include <sys/thread.h>
  58 #include <sys/x_call.h>
  59 #include <sys/smp_impldefs.h>
  60 #include <vm/vm_dep.h>
  61 #include <sys/psm.h>
```



```
  62 #include <sys/epm.h>
  63 #include <sys/cpr_wakecode.h>
  64 #include <sys/x86_archext.h>
  65 #include <sys/reboot.h>
  66 #include <acpica/include/acpi.h>
  66 #include <sys/acpi/acpi.h>
  67 #include <sys/acpica.h>
  68 #include <sys/fp.h>
  69 #include <sys/sysmacros.h>

  71 #define AFMT    "%lx"

  73 extern int      flushes_require_xcalls;
  74 extern cpuset_t cpu_ready_set;

  76 #if defined(__amd64)
  77 extern void     *wc_long_mode_64(void);
  78 #endif  /* __amd64 */
  79 extern int      tsc_gethrtime_enable;
  80 extern  void    i_cpr_start_cpu(void);

  82 ushort_t        cpr_mach_type = CPR_MACHTYPE_X86;
  83 void            (*cpr_start_cpu_func)(void) = i_cpr_start_cpu;

  85 static wc_cpu_t *wc_other_cpus = NULL;
  86 static cpuset_t procset;

  88 static void
  89 init_real_mode_platter(int cpun, uint32_t offset, uint_t cr4, wc_desctbr_t gdt);

  91 static int i_cpr_platform_alloc(psm_state_request_t *req);
  92 static void i_cpr_platform_free(psm_state_request_t *req);
  93 static int i_cpr_save_apic(psm_state_request_t *req);
  94 static int i_cpr_restore_apic(psm_state_request_t *req);
  95 static int wait_for_set(cpuset_t *set, int who);

  97 static  void i_cpr_save_stack(kthread_t *t, wc_cpu_t *wc_cpu);
  98 void i_cpr_restore_stack(kthread_t *t, greg_t *save_stack);

 100 #ifdef STACK_GROWTH_DOWN
 101 #define CPR_GET_STACK_START(t) ((t)->t_stkbase)
 102 #define CPR_GET_STACK_END(t) ((t)->t_stk)
 103 #else
 104 #define CPR_GET_STACK_START(t) ((t)->t_stk)
 105 #define CPR_GET_STACK_END(t) ((t)->t_stkbase)
 106 #endif  /* STACK_GROWTH_DOWN */

 108 /*
 109  * restart paused slave cpus
 110  */
 111 void
 112 i_cpr_machdep_setup(void)
 113 {
 114         if (ncpus > 1) {
 115                 CPR_DEBUG(CPR_DEBUG1, ("MP restarted...\n"));
 116                 mutex_enter(&cpu_lock);
 117                 start_cpus();
 118                 mutex_exit(&cpu_lock);
 119         }
 120 }
______unchanged_portion_omitted_
```

```
**********************************************************
   26724 Thu Dec 26 13:50:31 2013
new/usr/src/uts/i86pc/os/cpupm/cpu_idle.c
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */
  25 /*
  26  * Copyright (c) 2009-2010, Intel Corporation.
  27  * All rights reserved.
  28  */

  30 #include <sys/x86_archext.h>
  31 #include <sys/machsystm.h>
  32 #include <sys/x_call.h>
  33 #include <sys/stat.h>
  34 #include <acpica/include/acpi.h>
  34 #include <sys/acpi/acpi.h>
  35 #include <sys/acpica.h>
  36 #include <sys/cpu_acpi.h>
  37 #include <sys/cpu_idle.h>
  38 #include <sys/cpupm.h>
  39 #include <sys/cpu_event.h>
  40 #include <sys/hpet.h>
  41 #include <sys/archsystm.h>
  42 #include <vm/hat_i86.h>
  43 #include <sys/dtrace.h>
  44 #include <sys/sdt.h>
  45 #include <sys/callb.h>

  47 #define CSTATE_USING_HPET               1
  48 #define CSTATE_USING_LAT                2

  50 #define CPU_IDLE_STOP_TIMEOUT           1000

  52 extern void cpu_idle_adaptive(void);
  53 extern uint32_t cpupm_next_cstate(cma_c_state_t *cs_data,
  54     cpu_acpi_cstate_t *cstates, uint32_t cs_count, hrtime_t start);

  56 static int cpu_idle_init(cpu_t *);
  57 static void cpu_idle_fini(cpu_t *);
  58 static void cpu_idle_stop(cpu_t *);
  59 static boolean_t cpu_deep_idle_callb(void *arg, int code);
  60 static boolean_t cpu_idle_cpr_callb(void *arg, int code);
```



```
  61 static void acpi_cpu_cstate(cpu_acpi_cstate_t *cstate);

  63 static boolean_t cstate_use_timer(hrtime_t *lapic_expire, int timer);

  65 /*
  66  * the flag of always-running local APIC timer.
  67  * the flag of HPET Timer use in deep cstate.
  68  */
  69 static boolean_t cpu_cstate_arat = B_FALSE;
  70 static boolean_t cpu_cstate_hpet = B_FALSE;

  72 /*
  73  * Interfaces for modules implementing Intel's deep c-state.
  74  */
  75 cpupm_state_ops_t cpu_idle_ops = {
  76         "Generic ACPI C-state Support",
  77         cpu_idle_init,
  78         cpu_idle_fini,
  79         NULL,
  80         cpu_idle_stop
  81 };
______unchanged_portion_omitted_
```

```
**********************************************************
   26818 Thu Dec 26 13:50:32 2013
new/usr/src/uts/i86pc/os/cpupm/cpupm_mach.c
PANKOVs restructure
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */
  25 /*
  26  * Copyright (c) 2009, Intel Corporation.
  27  * All rights reserved.
  28  */

  30 #include <sys/cpu_pm.h>
  31 #include <sys/x86_archext.h>
  32 #include <sys/sdt.h>
  33 #include <sys/spl.h>
  34 #include <sys/machsystm.h>
  35 #include <sys/archsystm.h>
  36 #include <sys/hpet.h>
  37 #include <acpica/include/acpi.h>
  37 #include <sys/acpi/acpi.h>
  38 #include <sys/acpica.h>
  39 #include <sys/cpupm.h>
  40 #include <sys/cpu_idle.h>
  41 #include <sys/cpu_acpi.h>
  42 #include <sys/cpupm_throttle.h>
  43 #include <sys/dtrace.h>
  44 #include <sys/note.h>

  46 /*
  47  * This callback is used to build the PPM CPU domains once
  48  * a CPU device has been started. The callback is initialized
  49  * by the PPM driver to point to a routine that will build the
  50  * domains.
  51  */
  52 void (*cpupm_ppm_alloc_pstate_domains)(cpu_t *);

  54 /*
  55  * This callback is used to remove CPU from the PPM CPU domains
  56  * when the cpu driver is detached. The callback is initialized
  57  * by the PPM driver to point to a routine that will remove CPU
  58  * from the domains.
  59  */
  60 void (*cpupm_ppm_free_pstate_domains)(cpu_t *);
```



```
  62 /*
  63  * This callback is used to redefine the topspeed for a CPU device.
  64  * Since all CPUs in a domain should have identical properties, this
  65  * callback is initialized by the PPM driver to point to a routine
  66  * that will redefine the topspeed for all devices in a CPU domain.
  67  * This callback is exercised whenever an ACPI _PPC change notification
  68  * is received by the CPU driver.
  69  */
  70 void (*cpupm_redefine_topspeed)(void *);

  72 /*
  73  * This callback is used by the PPM driver to call into the CPU driver
  74  * to find a CPU's current topspeed (i.e., it's current ACPI _PPC value).
  75  */
  76 void (*cpupm_set_topspeed_callb)(void *, int);

  78 /*
  79  * This callback is used by the PPM driver to call into the CPU driver
  80  * to set a new topspeed for a CPU.
  81  */
  82 int (*cpupm_get_topspeed_callb)(void *);

  84 static void cpupm_event_notify_handler(ACPI_HANDLE, UINT32, void *);
  85 static void cpupm_free_notify_handlers(cpu_t *);
  86 static void cpupm_power_manage_notifications(void *);

  88 /*
  89  * Until proven otherwise, all power states are manageable.
  90  */
  91 static uint32_t cpupm_enabled = CPUPM_ALL_STATES;

  93 cpupm_state_domains_t *cpupm_pstate_domains = NULL;
  94 cpupm_state_domains_t *cpupm_tstate_domains = NULL;
  95 cpupm_state_domains_t *cpupm_cstate_domains = NULL;

  97 /*
  98  * c-state tunables
  99  *
 100  * cpupm_cs_sample_interval is the length of time we wait before
 101  * recalculating c-state statistics.  When a CPU goes idle it checks
 102  * to see if it has been longer than cpupm_cs_sample_interval since it last
 103  * caculated which C-state to go to.
 104  *
 105  * cpupm_cs_idle_cost_tunable is the ratio of time CPU spends executing + idle
 106  * divided by time spent in the idle state transitions.
 107  * A value of 10 means the CPU will not spend more than 1/10 of its time
 108  * in idle latency.  The worst case performance will be 90% of non Deep C-state
 109  * kernel.
 110  *
 111  * cpupm_cs_idle_save_tunable is how long we must stay in a deeper C-state
 112  * before it is worth going there.  Expressed as a multiple of latency.
 113  */
 114 uint32_t cpupm_cs_sample_interval = 100*1000*1000;      /* 100 milliseconds */
 115 uint32_t cpupm_cs_idle_cost_tunable = 10;       /* work time / latency cost */
 116 uint32_t cpupm_cs_idle_save_tunable = 2;        /* idle power savings */
 117 uint16_t cpupm_C2_idle_pct_tunable = 70;
 118 uint16_t cpupm_C3_idle_pct_tunable = 80;

 120 #ifndef __xpv
 121 extern boolean_t cpupm_intel_init(cpu_t *);
 122 extern boolean_t cpupm_amd_init(cpu_t *);

 124 typedef struct cpupm_vendor {
 125         boolean_t       (*cpuv_init)(cpu_t *);
 126 } cpupm_vendor_t;
______unchanged_portion_omitted_
```

```
**********************************************************
    7735 Thu Dec 26 13:50:32 2013
new/usr/src/uts/i86pc/os/cpupm/pwrnow.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/x86_archext.h>
#include <sys/machsystm.h>
#include <sys/x_call.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/pwrnow.h>
#include <sys/cpu_acpi.h>
#include <sys/cpupm.h>
#include <sys/dtrace.h>
#include <sys/sdt.h>

static int pwrnow_init(cpu_t *);
static void pwrnow_fini(cpu_t *);
static void pwrnow_power(cpuset_t, uint32_t);
static void pwrnow_stop(cpu_t *);

static boolean_t pwrnow_cpb_supported(void);

/*
 * Interfaces for modules implementing AMD's PowerNow!.
 */
cpupm_state_ops_t pwrnow_ops = {
        "PowerNow! Technology",
        pwrnow_init,
        pwrnow_fini,
        pwrnow_power,
        pwrnow_stop
};
______unchanged_portion_omitted_
```

```
**********************************************************
    7884 Thu Dec 26 13:50:33 2013
new/usr/src/uts/i86pc/os/cpupm/speedstep.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2009,  Intel Corporation.
 * All Rights Reserved.
 */

#include <sys/x86_archext.h>
#include <sys/machsystm.h>
#include <sys/archsystm.h>
#include <sys/x_call.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/speedstep.h>
#include <sys/cpu_acpi.h>
#include <sys/cpupm.h>
#include <sys/dtrace.h>
#include <sys/sdt.h>

static int speedstep_init(cpu_t *);
static void speedstep_fini(cpu_t *);
static void speedstep_power(cpuset_t, uint32_t);
static void speedstep_stop(cpu_t *);
static boolean_t speedstep_turbo_supported(void);

/*
 * Interfaces for modules implementing Intel's Enhanced SpeedStep.
 */
cpupm_state_ops_t speedstep_ops = {
        "Enhanced SpeedStep Technology",
        speedstep_init,
        speedstep_fini,
        speedstep_power,
        speedstep_stop
};
_____unchanged_portion_omitted_
```

```
**********************************************************
    5163 Thu Dec 26 13:50:33 2013
new/usr/src/uts/i86pc/os/cpupm/turbo.c
PANKOVs restructure
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2009,  Intel Corporation.
 * All Rights Reserved.
 */

#include <sys/x86_archext.h>
#include <sys/machsystm.h>
#include <sys/archsystm.h>
#include <sys/x_call.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/speedstep.h>
#include <sys/cpu_acpi.h>
#include <sys/cpupm.h>
#include <sys/dtrace.h>
#include <sys/sdt.h>

typedef struct turbo_kstat_s {
        struct kstat_named      turbo_supported;        /* turbo flag */
        struct kstat_named      t_mcnt;                 /* IA32_MPERF_MSR */
        struct kstat_named      t_acnt;                 /* IA32_APERF_MSR */
} turbo_kstat_t;
______unchanged_portion_omitted_
```

```
**********************************************************
   19844 Thu Dec 26 13:50:33 2013
new/usr/src/uts/i86pc/sys/acpidev.h
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#ifndef _SYS_ACPIDEV_H
#define _SYS_ACPIDEV_H
#include <sys/types.h>
#include <sys/obpdefs.h>
#include <sys/sunddi.h>
#ifdef  _KERNEL
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

/* Maximum recursion levels when enumerating objects in ACPI namespace. */
#define ACPIDEV_MAX_ENUM_LEVELS         32

/* Maximum length of device name for ACPI object. */
#define ACPIDEV_MAX_NAMELEN             OBP_MAXDRVNAME

/* Pseudo ACPI device HID for ACPI root object. */
#define ACPIDEV_HID_ROOTNEX             "SOLA0001"
/* Pseudo ACPI device HID for ACPI virtual bus. */
#define ACPIDEV_HID_VIRTNEX             "SOLA0002"
#define ACPIDEV_HID_SCOPE               "SOLA0003"
#define ACPIDEV_HID_PROCESSOR           "SOLA0004"

/* ACPI device HIDs/CIDs defined by ACPI specification. */
#define ACPIDEV_HID_CONTAINER1          "PNP0A05"
#define ACPIDEV_HID_CONTAINER2          "PNP0A06"
#define ACPIDEV_HID_MODULE              "ACPI0004"
#define ACPIDEV_HID_CPU                 "ACPI0007"
#define ACPIDEV_HID_PCI_HOSTBRIDGE      "PNP0A03"
#define ACPIDEV_HID_PCIE_HOSTBRIDGE     "PNP0A08"
```



```
#define ACPIDEV_HID_PCIEX_HOSTBRIDGE    "PNP0A08"
#define ACPIDEV_HID_MEMORY              "PNP0C80"

/* Device names for ACPI objects. */
#define ACPIDEV_NODE_NAME_ROOT          "fw"
#define ACPIDEV_NODE_NAME_ACPIDR        "acpidr"
#define ACPIDEV_NODE_NAME_CONTAINER     "container"
#define ACPIDEV_NODE_NAME_MODULE_SBD    "sb"
#define ACPIDEV_NODE_NAME_MODULE_CPU    "socket"
#define ACPIDEV_NODE_NAME_CPU           "cpu"
#define ACPIDEV_NODE_NAME_PROCESSOR     "cpus"
#define ACPIDEV_NODE_NAME_MEMORY        "mem"
#define ACPIDEV_NODE_NAME_PCI           "pci"

/* Device types for ACPI objects. */
#define ACPIDEV_TYPE_ROOTNEX            "acpirootnex"
#define ACPIDEV_TYPE_VIRTNEX            "acpivirtnex"
#define ACPIDEV_TYPE_SCOPE              "acpiscope"
#define ACPIDEV_TYPE_DEVICE             "acpidevice"
#define ACPIDEV_TYPE_CONTAINER          "acpicontainer"
#define ACPIDEV_TYPE_CPU                "acpicpu"
#define ACPIDEV_TYPE_MEMORY             "acpimemory"
#define ACPIDEV_TYPE_PCI                "pci"
#define ACPIDEV_TYPE_PCIEX              "pciex"

/* Device property names for ACPI objects. */
#define ACPIDEV_PROP_NAME_UNIT_ADDR     "unit-address"
#define ACPIDEV_PROP_NAME_ACPI_UID      "acpi-uid"
#define ACPIDEV_PROP_NAME_PROCESSOR_ID  "acpi-processor-id"
#define ACPIDEV_PROP_NAME_LOCALAPIC_ID  "apic-id"
#define ACPIDEV_PROP_NAME_PROXIMITY_ID  "proximity-id"

#define ACPIDEV_PROP_NAME_UID_FORMAT    "acpidev-uid-format"

/* Miscellaneous strings. */
#define ACPIDEV_CMD_OST_PREFIX          "acpi-update-status"
#define ACPIDEV_CMD_OST_INPROGRESS      "acpi-update-status=inprogress"
#define ACPIDEV_CMD_OST_SUCCESS         "acpi-update-status=success"
#define ACPIDEV_CMD_OST_FAILURE         "acpi-update-status=failure"
#define ACPIDEV_CMD_OST_NOOP            "acpi-update-status=noop"

#define ACPIDEV_EVENT_TYPE_ATTR_NAME    "acpi-event-type"
#define ACPIDEV_EVENT_TYPE_BUS_CHECK    "bus_check"
#define ACPIDEV_EVENT_TYPE_DEVICE_CHECK "device_check"
#define ACPIDEV_EVENT_TYPE_DEVICE_CHECK_LIGHT   "device_check_light"
#define ACPIDEV_EVENT_TYPE_EJECT_REQUEST        "eject_request"

/* ACPI device class Id. */
typedef enum acpidev_class_id {
        ACPIDEV_CLASS_ID_INVALID = 0,
        ACPIDEV_CLASS_ID_ROOTNEX = 1,
        ACPIDEV_CLASS_ID_SCOPE = 2,
        ACPIDEV_CLASS_ID_DEVICE = 3,
        ACPIDEV_CLASS_ID_CONTAINER = 4,
        ACPIDEV_CLASS_ID_CPU = 5,
        ACPIDEV_CLASS_ID_MEMORY = 6,
        ACPIDEV_CLASS_ID_PCI = 7,
        ACPIDEV_CLASS_ID_PCIEX = 8,
        ACPIDEV_CLASS_ID_MAX
} acpidev_class_id_t;

/* Flags for acpidev_options boot options. */
#define ACPIDEV_OUSER_NO_CPU            0x1
#define ACPIDEV_OUSER_NO_MEM            0x2
#define ACPIDEV_OUSER_NO_CONTAINER      0x4
#define ACPIDEV_OUSER_NO_PCI            0x8
```

```
126 #define ACPIDEV_OUSER_NO_CACHE          0x10000

128 #ifdef  _KERNEL

130 /* Common ACPI object names. */
131 #define ACPIDEV_OBJECT_NAME_SB          METHOD_NAME__SB_
131 #define ACPIDEV_OBJECT_NAME_SB          ACPI_NS_SYSTEM_BUS
132 #define ACPIDEV_OBJECT_NAME_PR          "_PR_"

134 /* Common ACPI method names. */
135 #define ACPIDEV_METHOD_NAME_MAT         "_MAT"
136 #define ACPIDEV_METHOD_NAME_EJ0         "_EJ0"
137 #define ACPIDEV_METHOD_NAME_EDL         "_EDL"
138 #define ACPIDEV_METHOD_NAME_EJD         "_EJD"
139 #define ACPIDEV_METHOD_NAME_OST         "_OST"
140 #define ACPIDEV_METHOD_NAME_PXM         "_PXM"
141 #define ACPIDEV_METHOD_NAME_SLI         "_SLI"

143 /* Source event code for _OST. */
144 #define ACPI_OST_EVENT_EJECTING         0x103
145 #define ACPI_OST_EVENT_INSERTING        0x200

147 /* Status code for _OST. */
148 #define ACPI_OST_STA_SUCCESS            0x0

150 /* Non-specific failure. */
151 #define ACPI_OST_STA_FAILURE            0x1

153 /* Unrecognized Notify Code. */
154 #define ACPI_OST_STA_NOT_SUPPORT        0x2

156 /* Device ejection not supported by OSPM. */
157 #define ACPI_OST_STA_EJECT_NOT_SUPPORT  0x80

159 /* Device in use by application. */
160 #define ACPI_OST_STA_EJECT_IN_USE       0x81

162 /* Device Busy. */
163 #define ACPI_OST_STA_EJECT_BUSY         0x82

165 /* Ejection dependency is busy or not supported for ejection by OSPM. */
166 #define ACPI_OST_STA_EJECT_DEPENDENCY   0x83

168 /* Ejection is in progress (pending). */
169 #define ACPI_OST_STA_EJECT_IN_PROGRESS  0x84

171 /* Device insertion in progress (pending). */
172 #define ACPI_OST_STA_INSERT_IN_PROGRESS 0x80

174 /* Device driver load failure. */
175 #define ACPI_OST_STA_INSERT_DRIVER      0x81

177 /* Device insertion not supported by OSPM. */
178 #define ACPI_OST_STA_INSERT_NOT_SUPPORT 0x82

180 /*
181  * Insertion failure
182  * Resources Unavailable as described by the following bit encodings:
183  * Bit[3] Bus Numbers
184  * Bit[2] Interrupts
185  * Bit[1] I/O
186  * Bit[0] Memory
187  */
188 #define ACPI_OST_STA_INSERT_NO_RESOURCE 0x90
189 #define ACPI_OST_STA_INSERT_NO_BUS      0x8
190 #define ACPI_OST_STA_INSERT_NO_INTR     0x4
```



```
191 #define ACPI_OST_STA_INSERT_NO_IO       0x2
192 #define ACPI_OST_STA_INSERT_NO_MEM      0x1

194 /*
195  * According to the ACPI specification, self latency (entry[n][n]) in the
196  * SLIT table should be 10.
197  */
198 #define ACPI_SLIT_SELF_LATENCY          10

200 /*
201  * The DR driver assigns a unique device id for each hot-added memory device.
202  * ACPI_MEMNODE_DEVID_BOOT is assigned to memory devices present at boot,
203  * which is distinguished from device ids assigned by the DR driver.
204  */
205 #define ACPI_MEMNODE_DEVID_BOOT         UINT32_MAX

207 /* Forward declaration */
208 typedef struct acpidev_data_impl        *acpidev_data_handle_t;
209 typedef struct acpidev_walk_info        acpidev_walk_info_t;
210 typedef struct acpidev_filter_rule      acpidev_filter_rule_t;
211 typedef struct acpidev_class            acpidev_class_t;
212 typedef struct acpidev_class_list       acpidev_class_list_t;

214 /* Type of ACPI device enumerating operation. */
215 typedef enum acpidev_op_type {
216         ACPIDEV_OP_BOOT_PROBE = 0,      /* First pass probing at boot time. */
217         ACPIDEV_OP_BOOT_REPROBE,        /* Second pass probing at boot time. */
218         ACPIDEV_OP_HOTPLUG_PROBE        /* Probing for hotplug at runtime. */
219 } acpidev_op_type_t;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    8327 Thu Dec 26 13:50:34 2013
new/usr/src/uts/i86pc/sys/acpidev_dr.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Interfaces to support System Board Dynamic Reconfiguration.
 */

#ifndef _SYS_ACPIDEV_DR_H
#define _SYS_ACPIDEV_DR_H
#include <sys/types.h>
#include <sys/obpdefs.h>
#include <sys/cpuvar.h>
#include <sys/memlist.h>
#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_rsc.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifdef  _KERNEL

/* Maximum number of DR capable system boards supported. */
#define ACPIDEV_DR_MAX_BOARDS           0x40
#define ACPIDEV_DR_SEGS_PER_MEM_DEV     0x10
#define ACPIDEV_DR_MEMLISTS_PER_SEG     0x10
#define ACPIDEV_DR_MAX_MEMLIST_ENTRIES  0x10000

#define ACPIDEV_DR_PROP_PORTID          "portid"
#define ACPIDEV_DR_PROP_BOARDNUM        OBP_BOARDNUM
#define ACPIDEV_DR_PROP_DEVNAME         OBP_NAME

/*
 * Format strings for DR capable system boards.
 * They will be used as attachment point names.
```



```
 */
#define ACPIDEV_DR_CPU_BD_FMT           "CPU%u"
#define ACPIDEV_DR_MEMORY_BD_FMT        "MEM%u"
#define ACPIDEV_DR_IO_BD_FMT            "IO%u"
#define ACPIDEV_DR_SYSTEM_BD_FMT        "SB%u"

typedef enum {
        ACPIDEV_INVALID_BOARD = 0,
        ACPIDEV_CPU_BOARD,
        ACPIDEV_MEMORY_BOARD,
        ACPIDEV_IO_BOARD,
        ACPIDEV_SYSTEM_BOARD
} acpidev_board_type_t;
______unchanged_portion_omitted_
```

```
**********************************************************
    4564 Thu Dec 26 13:50:34 2013
new/usr/src/uts/i86pc/sys/acpidev_impl.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

#ifndef _SYS_ACPIDEV_IMPL_H
#define _SYS_ACPIDEV_IMPL_H
#include <sys/types.h>
#include <sys/cmn_err.h>
#include <sys/bitmap.h>
#include <sys/synch.h>
#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/acpidev.h>
#include <sys/acpidev_dr.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifdef  _KERNEL

#define ACPIDEV_ARRAY_PARAM(a)          (a), (sizeof (a) / sizeof ((a)[0]))

/* Debug support facilities. */
extern int acpidev_debug;
#define ACPIDEV_DEBUG(lvl, ...) if (acpidev_debug) cmn_err((lvl), __VA_ARGS__)

/* Data attached to an ACPI object to maintain device status information. */
struct acpidev_data_impl {
        uint32_t                        aod_eflag;      /* External flags */
        uint32_t                        aod_iflag;      /* Internal flags */
        uint32_t                        aod_level;
        int                             aod_status;     /* Cached _STA value */
        ACPI_HANDLE                     *aod_hdl;
        dev_info_t                      *aod_dip;
        acpidev_class_t                 *aod_class;
        acpidev_class_list_t            **aod_class_list;
        acpidev_board_type_t            aod_bdtype;     /* Type of board. */
```



```
        uint32_t                        aod_bdnum;      /* Board # for DR. */
        uint32_t                        aod_portid;     /* Port id for DR. */
        uint32_t                        aod_bdidx;      /* Index # of AP */
        volatile uint32_t               aod_chidx;      /* Index # of child */
        uint32_t                        aod_memidx;     /* Index # of memory */
        acpidev_class_id_t              aod_class_id;   /* Dev type for DR. */
};
______unchanged_portion_omitted_
```

```
**********************************************************
    7693 Thu Dec 26 13:50:35 2013
new/usr/src/uts/i86pc/sys/acpidev_rsc.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009, Intel Corporation.
 * All rights reserved.
 */

#ifndef _SYS_ACPIDEV_RSC_H
#define _SYS_ACPIDEV_RSC_H
#include <sys/types.h>
#include <sys/obpdefs.h>
#include <sys/sunddi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

#ifdef __cplusplus
extern "C" {
#endif

/* ACPI bus range structure. */
typedef struct acpidev_bus_range {
        uint_t  bus_start;
        uint_t  bus_end;
} acpidev_bus_range_t;
```

*unchanged_portion_omitted_*

****
7156 Thu Dec 26 13:50:35 2013
new/usr/src/uts/i86pc/sys/cpu_acpi.h
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _CPU_ACPI_H
#define _CPU_ACPI_H

#include <sys/cpuvar.h>

#include <acpica/include/acpi.h>
#include <acpica/include/accommon.h>
#include <acpica/include/acresrc.h>
#include <sys/acpi/acpi.h>
#include <sys/acpi/accommon.h>
#include <sys/acpi/acresrc.h>
#include <sys/acpica.h>

#ifdef __cplusplus
extern "C" {
#endif

/*
 * P-state related macros
 */
#define CPU_ACPI_PPC(sp)                sp->cs_ppc
#define CPU_ACPI_PSD(sp)                sp->cs_psd
#define CPU_ACPI_PCT(sp)                sp->cs_pct
#define CPU_ACPI_PCT_CTRL(sp)           &sp->cs_pct[0]
#define CPU_ACPI_PCT_STATUS(sp)         &sp->cs_pct[1]
#define CPU_ACPI_PSTATES(sp)            sp->cs_pstates.ss_states
#define CPU_ACPI_PSTATES_COUNT(sp)      sp->cs_pstates.ss_count

#define CPU_ACPI_FREQ(pstate)           pstate->ps_freq
#define CPU_ACPI_PSTATE_TRANSLAT(pstate) pstate->ps_translat
#define CPU_ACPI_PSTATE_CTRL(pstate)    pstate->ps_ctrl

/*
 * T-state related macros
 */
#define CPU_ACPI_TPC(sp)                sp->cs_tpc
```



```
#define CPU_ACPI_TSD(sp)                sp->cs_tsd
#define CPU_ACPI_PTC(sp)                sp->cs_ptc
#define CPU_ACPI_PTC_CTRL(sp)           &sp->cs_ptc[0]
#define CPU_ACPI_PTC_STATUS(sp)         &sp->cs_ptc[1]
#define CPU_ACPI_TSTATES(sp)            sp->cs_tstates.ss_states
#define CPU_ACPI_TSTATES_COUNT(sp)      sp->cs_tstates.ss_count

#define CPU_ACPI_FREQPER(tstate)        tstate->ts_freqper
#define CPU_ACPI_TSTATE_TRANSLAT(tstate) tstate->ts_translat
#define CPU_ACPI_TSTATE_CTRL(tstate)    tstate->ts_ctrl
#define CPU_ACPI_TSTATE_STAT(tstate)    tstate->ts_state

/*
 * C-state realted macros
 */
#define CPU_ACPI_CSD(sp)                sp->cs_csd
#define CPU_ACPI_CSTATES(sp)            sp->cs_cstates.ss_states
#define CPU_ACPI_CSTATES_COUNT(sp)      sp->cs_cstates.ss_count

#define CPU_ACPI_NONE_CACHED            0x0000
#define CPU_ACPI_PCT_CACHED             0x0001
#define CPU_ACPI_PSS_CACHED             0x0002
#define CPU_ACPI_PSD_CACHED             0x0004
#define CPU_ACPI_PPC_CACHED             0x0008
#define CPU_ACPI_PTC_CACHED             0x0010
#define CPU_ACPI_TSS_CACHED             0x0020
#define CPU_ACPI_TSD_CACHED             0x0040
#define CPU_ACPI_TPC_CACHED             0x0080
#define CPU_ACPI_CST_CACHED             0x0100
#define CPU_ACPI_CSD_CACHED             0x0200

#define CPU_ACPI_IS_OBJ_CACHED(sp, obj) (sp->cpu_acpi_cached & obj)
#define CPU_ACPI_OBJ_IS_CACHED(sp, obj) (sp->cpu_acpi_cached |= obj)
#define CPU_ACPI_OBJ_IS_NOT_CACHED(sp, obj) (sp->cpu_acpi_cached &= ~obj)

#define CPU_ACPI_PSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_pstate_t))
#define CPU_ACPI_PSS_CNT (sizeof (cpu_acpi_pstate_t) / sizeof (uint32_t))
#define CPU_ACPI_TSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_tstate_t))
#define CPU_ACPI_TSS_CNT (sizeof (cpu_acpi_tstate_t) / sizeof (uint32_t))
#define CPU_ACPI_CSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_cstate_t))
#define CPU_ACPI_CST_CNT (sizeof (cpu_acpi_cstate_t) / sizeof (uint32_t))
/*
 * CPU Domain Coordination Types
 */
#define CPU_ACPI_SW_ALL 0xfc
#define CPU_ACPI_SW_ANY 0xfd
#define CPU_ACPI_HW_ALL 0xfe

/*
 * Container for ACPI processor state dependency information
 */
typedef struct cpu_acpi_state_dependency
{
        uint8_t sd_entries;
        uint8_t sd_revision;
        uint32_t sd_domain;
        uint32_t sd_type;
        uint32_t sd_num;
        uint32_t sd_index;
} cpu_acpi_state_dependency_t;
```

unchanged_portion_omitted_

```
**********************************************************
    9810 Thu Dec 26 13:50:35 2013
new/usr/src/uts/i86pc/sys/hpet_acpi.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#ifndef _HPET_ACPI_H
#define _HPET_ACPI_H

#if defined(_KERNEL)
#include <acpica/include/acpi.h>
#include <acpica/include/actbl1.h>
#include <sys/acpi/acpi.h>
#include <sys/acpi/actbl1.h>
#include <sys/acpica.h>
#endif  /* defined(_KERNEL) */

#ifdef __cplusplus
extern "C" {
#endif

/*
 * Solaris uses an HPET Timer to generate interrupts for CPUs in Deep C-state
 * with stalled LAPIC Timers.  All CPUs use one HPET timer.  The timer's
 * interrupt targets one CPU (via the I/O APIC).  The one CPU that receives
 * the HPET's interrupt wakes up other CPUs as needed during the HPET Interrupt
 * Service Routing.  The HPET ISR uses poke_cpus to wake up other CPUs with an
 * Inter Processor Interrupt.
 *
 * Please see the Intel Programmer's guides.  Interrupts are disabled before
 * a CPU Halts into Deep C-state.  (This allows CPU-hardware-specific cleanup
 * before servicing interrupts.)  When a Deep C-state CPU wakes up (due to
 * an externally generated interrupt), it resume execution where it halted.
 * The CPU returning from Deep C-state must enable interrupts before it will
 * handle the pending interrupt that woke it from Deep C-state.
 *
 *
 * HPET bits as defined in the Intel IA-PC HPET Specification Rev 1.0a.
 *
 * The physical address space layout of the memory mapped HPET looks like this:
 *
 * struct hpet {
 *      uint64_t        gen_cap;
```



```
 *      uint64_t        res1;
 *      uint64_t        gen_config;
 *      uint64_t        res2;
 *      uint64_t        gen_inter_stat;
 *      uint64_t        res3;
 *      uint64_t        main_counter_value;
 *      uint64_t        res4;
 *      stuct hpet_timer {
 *              uint64_t        config_and_capability;
 *              uint64_t        comparator_value;
 *              uint64_t        FSB_interrupt_route;
 *              uint64_t        reserved;
 *      } timers[32];
 * }
 *
 * There are 32 possible timers in an hpet.  Only the first 3 timers are
 * required.  The other 29 timers are optional.
 *
 * HPETs can have 64-bit or 32-bit timers.  Timers/compare registers can
 * be 64-bit or 32-bit and can be a mixture of both.
 * The first two timers are not used.  The HPET spec intends the first two
 * timers to be used as "legacy replacement" for the PIT and RTC timers.
 *
 * Solaris uses the first available non-legacy replacement timer as a proxy
 * timer for processor Local APIC Timers that stop in deep idle C-states.
 */

/*
 * We only use HPET table 1 on x86.  Typical x86 systems only have 1 HPET.
 * ACPI allows for multiple HPET tables to describe multiple HPETs.
 */
#define HPET_TABLE_1            (1)

/*
 * HPET Specification 1.0a defines the HPET to occupy 1024 bytes regardless of
 * the number of counters (3 to 32) in this implementation.
 */
#define HPET_SIZE               (1024)

/*
 * Offsets of hpet registers and macros to access them from HPET base address.
 */
#define HPET_GEN_CAP_OFFSET             (0)
#define HPET_GEN_CONFIG_OFFSET          (0x10)
#define HPET_GEN_INTR_STAT_OFFSET       (0x20)
#define HPET_MAIN_COUNTER_OFFSET        (0xF0)
#define HPET_TIMER_N_CONF_OFFSET(n)     (0x100 + (n * 0x20))
#define HPET_TIMER_N_COMP_OFFSET(n)     (0x108 + (n * 0x20))

#define OFFSET_ADDR(a, o)               (((uintptr_t)(a)) + (o))
#define HPET_GEN_CAP_ADDRESS(la)                                        \
                    OFFSET_ADDR(la, HPET_GEN_CAP_OFFSET)
#define HPET_GEN_CONFIG_ADDRESS(la)                                     \
                    OFFSET_ADDR(la, HPET_GEN_CONFIG_OFFSET)
#define HPET_GEN_INTR_STAT_ADDRESS(la)                                  \
                    OFFSET_ADDR(la, HPET_GEN_INTR_STAT_OFFSET)
#define HPET_MAIN_COUNTER_ADDRESS(la)                                   \
                    OFFSET_ADDR(la, HPET_MAIN_COUNTER_OFFSET)
#define HPET_TIMER_N_CONF_ADDRESS(la, n)                                \
                    OFFSET_ADDR(la, HPET_TIMER_N_CONF_OFFSET(n))
#define HPET_TIMER_N_COMP_ADDRESS(la, n)                                \
                    OFFSET_ADDR(la, HPET_TIMER_N_COMP_OFFSET(n))

/*
 * HPET General Capabilities and ID Register
 */
```

```
 126 typedef struct hpet_gen_cap {
 127         uint32_t        counter_clk_period;     /* period in femtoseconds */
 128         uint32_t        vendor_id       :16;    /* vendor */
 129         uint32_t        leg_route_cap   :1;     /* 1=LegacyReplacemnt support */
 130         uint32_t        res1            :1;     /* reserved */
 131         uint32_t        count_size_cap  :1;     /* 0=32bit, 1=64bit wide */
 132         uint32_t        num_tim_cap     :5;     /* number of timers -1 */
 133         uint32_t        rev_id          :8;     /* revision number */
 134 } hpet_gen_cap_t;
```

_____***unchanged_portion_omitted_***

****
4042 Thu Dec 26 13:50:36 2013
new/usr/src/uts/i86pc/sys/psm_common.h
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

#ifndef _SYS_PSM_COMMON_H
#define _SYS_PSM_COMMON_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/types.h>
#include <sys/param.h>
#include <sys/cmn_err.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

#include <sys/sunddi.h>
#include <sys/ddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/debug.h>

#ifdef  __cplusplus
extern "C" {
#endif


/* private data used in psm_common */
typedef struct acpi_prs_private {
        uchar_t         prs_irqflags;
        uchar_t         prs_type;
} acpi_prs_private_t;
_____unchanged_portion_omitted_
```

```
**********************************************************
   62027 Thu Dec 26 13:50:36 2013
new/usr/src/uts/i86xpv/io/psm/mp_platform_xpv.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/apic_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
```



```
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/cyclic.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/sunndi.h>


/*
 *      Local Function Prototypes
 */
static void apic_mark_vector(uchar_t oldvector, uchar_t newvector);
static void apic_xlate_vector_free_timeout_handler(void *arg);
static int apic_check_stuck_interrupt(apic_irq_t *irq_ptr, int old_bind_cpu,
    int new_bind_cpu, int apicindex, int intin_no, int which_irq,
    struct ioapic_reprogram_data *drep);
static int apic_setup_irq_table(dev_info_t *dip, int irqno,
    struct apic_io_intr *intrp, struct intrspec *ispec, iflag_t *intr_flagp,
    int type);
static void apic_try_deferred_reprogram(int ipl, int vect);
static void delete_defer_repro_ent(int which_irq);
static void apic_ioapic_wait_pending_clear(int ioapicindex,
    int intin_no);

extern int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
    int ipin, int *pci_irqp, iflag_t *intr_flagp);
extern int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
    int child_ipin, struct apic_io_intr **intrp);
extern uchar_t acpi_find_ioapic(int irq);
extern struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
extern int apic_find_bus_id(int bustype);
extern int apic_find_intin(uchar_t ioapic, uchar_t intin);
extern void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

extern  int apic_sci_vect;
extern  iflag_t apic_sci_flags;
extern  int     apic_intr_policy;
extern  char *psm_name;

/*
 * number of bits per byte, from <sys/param.h>
 */
#define UCHAR_MAX       ((1 << NBBY) - 1)

/* Max wait time (in repetitions) for flags to clear in an RDT entry. */
extern int apic_max_reps_clear_pending;

/* The irq # is implicit in the array index: */
struct ioapic_reprogram_data apic_reprogram_info[APIC_MAX_VECTOR+1];
/*
 * APIC_MAX_VECTOR + 1 is the maximum # of IRQs as well. ioapic_reprogram_info
 * is indexed by IRQ number, NOT by vector number.
 */

extern  int     apic_int_busy_mark;
extern  int     apic_int_free_mark;
extern  int     apic_diff_for_redistribution;
extern  int     apic_sample_factor_redistribution;
extern  int     apic_redist_cpu_skip;
extern  int     apic_num_imbalance;
extern  int     apic_num_rebind;

/* timeout for xlate_vector, mark_vector */
```

```
127 int     apic_revector_timeout = 16 * 10000; /* 160 millisec */

129 extern int      apic_defconf;
130 extern int      apic_irq_translate;

132 extern int      apic_use_acpi_madt_only;        /* 1=ONLY use MADT from ACPI */

134 extern  uchar_t apic_io_vectbase[MAX_IO_APIC];

136 extern  boolean_t ioapic_mask_workaround[MAX_IO_APIC];

138 /*
139  * First available slot to be used as IRQ index into the apic_irq_table
140  * for those interrupts (like MSI/X) that don't have a physical IRQ.
141  */
142 extern int apic_first_avail_irq;

144 /*
145  * apic_defer_reprogram_lock ensures that only one processor is handling
146  * deferred interrupt programming at *_intr_exit time.
147  */
148 static  lock_t  apic_defer_reprogram_lock;

150 /*
151  * The current number of deferred reprogrammings outstanding
152  */
153 uint_t  apic_reprogram_outstanding = 0;

155 #ifdef DEBUG
156 /*
157  * Counters that keep track of deferred reprogramming stats
158  */
159 uint_t  apic_intr_deferrals = 0;
160 uint_t  apic_intr_deliver_timeouts = 0;
161 uint_t  apic_last_ditch_reprogram_failures = 0;
162 uint_t  apic_deferred_setup_failures = 0;
163 uint_t  apic_defer_repro_total_retries = 0;
164 uint_t  apic_defer_repro_successes = 0;
165 uint_t  apic_deferred_spurious_enters = 0;
166 #endif

168 extern  int     apic_io_max;
169 extern  struct apic_io_intr *apic_io_intrp;

171 uchar_t apic_vector_to_irq[APIC_MAX_VECTOR+1];

173 extern  uint32_t        eisa_level_intr_mask;
174         /* At least MSB will be set if EISA bus */

176 extern  int     apic_pci_bus_total;
177 extern  uchar_t apic_single_pci_busid;

179 /*
180  * Following declarations are for revectoring; used when ISRs at different
181  * IPLs share an irq.
182  */
183 static  lock_t  apic_revector_lock;
184 int     apic_revector_pending = 0;
185 static  uchar_t *apic_oldvec_to_newvec;
186 static  uchar_t *apic_newvec_to_oldvec;

188 /* ACPI Interrupt Source Override Structure ptr */
189 ACPI_MADT_INTERRUPT_OVERRIDE *acpi_isop;
190 extern  int acpi_iso_cnt;

192 /*
```



```
193  * Auto-configuration routines
194  */

196 /*
197  * Initialise vector->ipl and ipl->pri arrays. level_intr and irqtable
198  * are also set to NULL. vector->irq is set to a value which cannot map
199  * to a real irq to show that it is free.
200  */
201 void
202 apic_init_common(void)
203 {
204         int     i, j, indx;
205         int     *iptr;

207         /*
208          * Initialize apic_ipls from apic_vectortoipl.  This array is
209          * used in apic_intr_enter to determine the IPL to use for the
210          * corresponding vector.  On some systems, due to hardware errata
211          * and interrupt sharing, the IPL may not correspond to the IPL listed
212          * in apic_vectortoipl (see apic_addspl and apic_delspl).
213          */
214         for (i = 0; i < (APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL); i++) {
215                 indx = i * APIC_VECTOR_PER_IPL;

217                 for (j = 0; j < APIC_VECTOR_PER_IPL; j++, indx++)
218                         apic_ipls[indx] = apic_vectortoipl[i];
219         }

221         /* cpu 0 is always up (for now) */
222         apic_cpus[0].aci_status = APIC_CPU_ONLINE | APIC_CPU_INTR_ENABLE;

224         iptr = (int *)&apic_irq_table[0];
225         for (i = 0; i <= APIC_MAX_VECTOR; i++) {
226                 apic_level_intr[i] = 0;
227                 *iptr++ = NULL;
228                 apic_vector_to_irq[i] = APIC_RESV_IRQ;

230                 /* These *must* be initted to B_TRUE! */
231                 apic_reprogram_info[i].done = B_TRUE;
232                 apic_reprogram_info[i].irqp = NULL;
233                 apic_reprogram_info[i].tries = 0;
234                 apic_reprogram_info[i].bindcpu = 0;
235         }

237         /*
238          * Allocate a dummy irq table entry for the reserved entry.
239          * This takes care of the race between removing an irq and
240          * clock detecting a CPU in that irq during interrupt load
241          * sampling.
242          */
243         apic_irq_table[APIC_RESV_IRQ] =
244             kmem_zalloc(sizeof (apic_irq_t), KM_SLEEP);

246         mutex_init(&airq_mutex, NULL, MUTEX_DEFAULT, NULL);
247 }
______unchanged_portion_omitted_
```

```
**********************************************************
    7388 Thu Dec 26 13:50:37 2013
new/usr/src/uts/intel/Makefile.files
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012, Joyent, Inc. All rights reserved.
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

#
#       This Makefile defines all file modules and build rules for the
# directory uts/intel and its children. These are the source files which
# are specific to x86 processor architectures.
#

#
#       Core (unix) objects
#
CORE_OBJS +=            \
        arch_kdi.o      \
        copy.o          \
        copy_subr.o     \
        cpc_subr.o      \
        ddi_arch.o      \
        ddi_i86.o       \
        ddi_i86_asm.o   \
        desctbls.o      \
        desctbls_asm.o  \
        exception.o     \
        float.o         \
        fmsmb.o         \
        fpu.o           \
        i86_subr.o      \
        lock_prim.o     \
        ovbcopy.o       \
        polled_io.o     \
        sseblk.o        \
        sundep.o        \
        swtch.o         \
        sysi86.o

#
```



```
# 64-bit multiply/divide compiler helper routines
# used only for ia32
#

SPECIAL_OBJS_32 +=      \
        muldiv.o

#
#       Generic-unix Module
#
GENUNIX_OBJS +=         \
        archdep.o       \
        getcontext.o    \
        install_utrap.o \
        lwp_private.o   \
        prom_enter.o    \
        prom_exit.o     \
        prom_panic.o    \
        sendsig.o       \
        syscall.o


#
#       PROM Routines
#
GENUNIX_OBJS +=         \
        prom_env.o      \
        prom_emul.o     \
        prom_getchar.o  \
        prom_init.o     \
        prom_node.o     \
        prom_printf.o   \
        prom_prop.o     \
        prom_putchar.o  \
        prom_reboot.o   \
        prom_version.o

#
#       file system modules
#
CORE_OBJS +=            \
        prmachdep.o

#
#       ZFS file system module
#
ZFS_OBJS +=             \
        spa_boot.o

#
#       Decompression code
#
CORE_OBJS += decompress.o

#
#       Microcode utilities
#
CORE_OBJS += ucode_utils.o

#
#       Driver modules
#
AGPGART_OBJS += agpgart.o agp_kstat.o
AGPTARGET_OBJS += agptarget.o
AMD64GART_OBJS += amd64_gart.o
ARCMSR_OBJS += arcmsr.o
```

```
ATA_OBJS += $(GHD_OBJS) ata_blacklist.o ata_common.o ata_disk.o \
        ata_dma.o atapi.o atapi_fsm.o ata_debug.o \
        sil3xxx.o
BSCBUS_OBJS += bscbus.o
BSCV_OBJS += bscv.o
CMDK_OBJS += cmdk.o
CMLB_OBJS += cmlb.o
CPUNEX_OBJS += cpunex.o
DADK_OBJS += dadk.o
DCOPY_OBJS += dcopy.o
DNET_OBJS += dnet.o dnet_mii.o
FD_OBJS += fd.o
GDA_OBJS += gda.o
GHD_OBJS += ghd.o ghd_debug.o ghd_dma.o ghd_queue.o ghd_scsa.o \
        ghd_scsi.o ghd_timer.o ghd_waitq.o ghd_gcmd.o
I915_OBJS += i915_dma.o i915_drv.o i915_irq.o i915_mem.o \
        i915_gem.o i915_gem_debug.o i915_gem_tiling.o
NSKERN_OBJS += nsc_asm.o
PCICFG_OBJS += pcicfg.o
PCI_PCINEXUS_OBJS += pci_pci.o
PCIEB_OBJS += pcieb_x86.o
PIT_BEEP_OBJS += pit_beep.o
POWER_OBJS += power.o
PCI_AUTOCONFIG_OBJS += pci_autoconfig.o pci_boot.o pcie_nvidia.o \
                       pci_memlist.o pci_resource.o
RADEON_OBJS += r300_cmdbuf.o radeon_cp.o radeon_drv.o \
                       radeon_state.o radeon_irq.o radeon_mem.o
SD_OBJS += sd.o sd_xbuf.o

HECI_OBJS +=            \
        heci_init.o     \
        heci_intr.o     \
        heci_interface.o \
        io_heci.o       \
        heci_main.o

STRATEGY_OBJS += strategy.o
UCODE_OBJS += ucode_drv.o
VGATEXT_OBJS += vgatext.o vgasubr.o

#
#       Kernel linker
#
KRTLD_OBJS +=           \
        bootrd.o        \
        ufsops.o        \
        hsfs.o          \
        doreloc.o       \
        kobj_boot.o     \
        kobj_convrelstr.o \
        kobj_crt.o      \
        kobj_isa.o      \
        kobj_reloc.o

#
#       ACPI CA module
#       misc. modules
#

# native
ACPICA_OBJS +=  acpica.o acpica_ec.o acpi_enum.o master_ops.o osl.o     \
                osl_ml.o
# debugger
ACPICA_OBJS +=  dbcmds.o dbdisply.o dbexec.o dbfileio.o dbhistry.o      \
                dbinput.o dbmethod.o dbnames.o dbstats.o dbutils.o      \
                dbxface.o dbconvert.o
```



```
# disassembler
ACPICA_OBJS +=  dmbuffer.o dmnames.o dmobject.o dmopcode.o dmresrc.o    \
                dmresrcl.o dmresrcs.o dmutils.o dmwalk.o dmdeferred.o   \
                dmresrcl2.o
# dispatcher
ACPICA_OBJS +=  dsargs.o dscontrol.o dsfield.o dsinit.o dsmethod.o      \
ACPICA_OBJS     += dbcmds.o dbdisply.o \
                   dbexec.o dbfileio.o dbhistry.o dbinput.o dbstats.o \
                   dbutils.o dbxface.o evevent.o evgpe.o evgpeblk.o \
                   evmisc.o evregion.o evrgnini.o evsci.o evxface.o \
                   evxfevnt.o evxfregn.o hwacpi.o hwgpe.o hwregs.o \
                   hwsleep.o hwtimer.o dsfield.o dsinit.o dsmethod.o \
                dsmthdat.o dsobject.o dsopcode.o dsutils.o dswexec.o    \
                dswload.o dswload2.o dswscope.o dswstate.o
# events
ACPICA_OBJS +=  evevent.o evglock.o evgpe.o evgpeblk.o evgpeinit.o      \
                evgpeutil.o evmisc.o evregion.o evrgnini.o evsci.o      \
                evxface.o evxfevnt.o evxfgpe.o evxfregn.o evhandler.o
# executer
ACPICA_OBJS +=  exconfig.o exconvrt.o excreate.o exdebug.o exdump.o     \
                exfield.o exfldio.o exmisc.o exmutex.o exnames.o        \
                exoparg1.o exoparg2.o exoparg3.o exoparg6.o exprep.o    \
                exregion.o exresnte.o exresolv.o exresop.o exstore.o    \
                exstoren.o exstorob.o exsystem.o exutils.o
# hardware
ACPICA_OBJS +=  hwacpi.o hwgpe.o hwpci.o hwregs.o hwsleep.o hwtimer.o   \
                hwvalid.o hwxface.o hwesleep.o hwxfsleep.o
# namespace
ACPICA_OBJS +=  nsaccess.o nsalloc.o nsdump.o nsdumpdv.o nseval.o       \
                nsinit.o nsload.o nsnames.o nsobject.o nsparse.o        \
                nspredef.o nsrepair.o nsrepair2.o nssearch.o nsutils.o  \
                nswalk.o nsxfeval.o nsxfname.o nsxfobj.o nsarguments.o  \
                nsconvert.o nsprepkg.o
# parser
ACPICA_OBJS +=  psargs.o psloop.o psopcode.o psparse.o psscope.o        \
                pstree.o psutils.o pswalk.o psxface.o psobject.o psopinfo.o
# resources
ACPICA_OBJS +=  rsaddr.o rscalc.o rscreate.o rsdump.o rsinfo.o rsio.o   \
                rsirq.o rslist.o rsmemory.o rsmisc.o rsutils.o          \
                rsxface.o rsdumpinfo.o rsserial.o
# tables
ACPICA_OBJS +=  tbfadt.o tbfind.o tbinstal.o tbutils.o tbxface.o        \
                tbxfroot.o tbprint.o tbxfload.o
# utilities
ACPICA_OBJS +=  utalloc.o utcache.o utclib.o utcopy.o utdebug.o         \
                utdecode.o utdelete.o uteval.o utglobal.o utids.o       \
                utinit.o utlock.o utmath.o utmisc.o utmutex.o           \
                utobject.o utosi.o utresrc.o utstate.o uttrack.o        \
                utxface.o utxferror.o utaddress.o utbuffer.o uterror.o  \
                utexcep.o utownerid.o utpredef.o utstring.o utxfinit.o  \
                utxfmutex.o
                   dswload.o dswscope.o dswstate.o exconfig.o exconvrt.o \
                   excreate.o exdump.o exfield.o exfldio.o exmisc.o \
                   exmutex.o exnames.o exoparg1.o exoparg2.o exoparg3.o \
                   exoparg6.o exprep.o exregion.o exresnte.o exresolv.o \
                   exresop.o exstore.o exstoren.o exstorob.o exsystem.o \
                   exutils.o psargs.o psopcode.o psparse.o psscope.o \
                   pstree.o psutils.o pswalk.o psxface.o nsaccess.o \
                   nsalloc.o nsdump.o nsdumpdv.o nseval.o nsinit.o \
                   nsload.o nsnames.o nsobject.o nsparse.o nssearch.o \
                   nsutils.o nswalk.o nsxfeval.o nsxfname.o nsxfobj.o \
                   rsaddr.o rscalc.o rscreate.o rsdump.o \
                   rsinfo.o rsio.o rsirq.o rslist.o rsmemory.o rsmisc.o \
                   rsutils.o rsxface.o tbfadt.o tbfind.o tbinstal.o  \
                   tbutils.o tbxface.o tbxfroot.o \
                   utalloc.o utclib.o utcopy.o utdebug.o utdelete.o \
```

```
205                     uteval.o utglobal.o utinit.o utmath.o utmisc.o \
206                     utobject.o utresrc.o utxface.o acpica.o acpi_enum.o \
207                     master_ops.o osl.o osl_ml.o acpica_ec.o utcache.o \
208                     utmutex.o utstate.o dmbuffer.o dmnames.o dmobject.o \
209                     dmopcode.o dmresrc.o dmresrcl.o dmresrcs.o dmutils.o \
210                     dmwalk.o psloop.o nspredef.o hwxface.o hwvalid.o \
211                     utlock.o utids.o nsrepair.o nsrepair2.o \
212                     dbmethod.o dbnames.o dsargs.o dscontrol.o dswload2.o \
213                     evglock.o evgpeinit.o evgpeutil.o evxfgpe.o exdebug.o \
214                     hwpci.o utdecode.o utosi.o utxferror.o

238 #
239 #       misc. modules
240 #

242 AGP_OBJS += agpmaster.o
243 FBT_OBJS += fbt.o
244 SDT_OBJS += sdt.o

246 #
247 #       AMD8111 NIC driver module
248 #
249 AMD8111S_OBJS += amd8111s_main.o amd8111s_hw.o

251 #
252 #       Pentium Performance Counter BackEnd module
253 #
254 P123_PCBE_OBJS = p123_pcbe.o

256 #
257 #       Pentium 4 Performance Counter BackEnd module
258 #
259 P4_PCBE_OBJS = p4_pcbe.o

261 #
262 #       AMD Opteron/Athlon64 Performance Counter BackEnd module
263 #
264 OPTERON_PCBE_OBJS = opteron_pcbe.o

266 #
267 #       Intel Core Architecture Performance Counter BackEnd module
268 #
269 CORE_PCBE_OBJS = core_pcbe.o

271 #
272 #       AMR module
273 #
274 AMR_OBJS = amr.o

276 #
277 #       IPMI module
278 #
279 IPMI_OBJS +=    ipmi_main.o ipmi.o ipmi_kcs.o

281 #
282 #       IOMMULIB module
283 #
284 IOMMULIB_OBJS = iommulib.o

286 #
287 #       Brand modules
288 #
289 SN1_BRAND_OBJS  =       sn1_brand.o sn1_brand_asm.o
290 S10_BRAND_OBJS  =       s10_brand.o s10_brand_asm.o

292 #
```



```
293 #       special files
294 #
295 MODSTUB_OBJ +=          \
296         modstubs.o

298 BOOTDEV_OBJS +=         \
299         bootdev.o

301 INC_PATH        += -I$(UTSBASE)/intel


304 CPR_INTEL_OBJS +=       cpr_intel.o

306 #
307 # AMD family 0xf memory controller module
308 #
309 include $(SRC)/common/mc/mc-amd/Makefile.mcamd
310 MCAMD_OBJS      += \
311         $(MCAMD_CMN_OBJS) \
312         mcamd_drv.o \
313         mcamd_dimmcfg.o \
314         mcamd_subr.o \
315         mcamd_pcicfg.o

317 #
318 # Intel Nehalem memory controller module
319 #
320 INTEL_NHM_OBJS += \
321         nhm_init.o \
322         mem_addr.o \
323         intel_nhmdrv.o \
324         nhm_pci_cfg.o \
325         dimm_topo.o \
326         intel_nhm.o

328 #
329 # Intel 5000/5100/5400/7300 chipset memory controller hub (MCH) module
330 #
331 INTEL_NB5000_OBJS += \
332         intel_nb5000.o \
333         intel_nbdrv.o \
334         dimm_addr.o \
335         nb_pci_cfg.o \
336         nb5000_init.o
```

```
**********************************************************
   13941 Thu Dec 26 13:50:37 2013
new/usr/src/uts/intel/Makefile.rules
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2012 Joyent, Inc.  All rights reserved.
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

#
#       This Makefile defines all file modules and build rules for the
# directory uts/intel and its children. These are the source files which
# are specific to the intel processor.
#
#       The following two-level ordering must be maintained in this file.
#         Lines are sorted first in order of decreasing specificity based on
#         the first directory component.  That is, sun4u rules come before
#         sparc rules come before common rules.
#
#         Lines whose initial directory components are equal are sorted
#         alphabetically by the remaining components.

#
# Need a way to distinguish between the ia32 and amd64 subdirs.
#
SUBARCH_DIR_32 = ia32
SUBARCH_DIR_64 = amd64
SUBARCH_DIR     = $(SUBARCH_DIR_$(CLASS))

#
#       Section 1a: C object build rules
#
$(OBJS_DIR)/%.o:                $(SRC)/common/fs/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/power/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/util/i386/%.s
```



```
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/brand/sn1/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/brand/solaris10/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/dtrace/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/dtrace/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/fs/proc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/ia32/ml/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/ia32/os/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/ia32/promif/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/ia32/syscall/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/debugger/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/debugger/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/disassembler/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/events/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/dispatcher/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/hardware/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/events/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/dispatcher/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/executer/%.c
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/executer/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/hardware/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/parser/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/namespace/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/namespace/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/parser/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/resources/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/resources/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/tables/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/tables/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/utilities/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/acpica/components/utilities/%.c
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/acpica/disassembler/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/agpgart/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/agpmaster/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/amd8111s/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/amr/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/drm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/hotplug/pcicfg/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/ipmi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/intel_nb5000/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```



```
$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/intel_nhm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(SRC)/common/mc/mc-amd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/mc-amd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/pci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/pciex/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dktp/controller/ata/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dktp/dcdev/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dktp/disk/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dktp/drvobj/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dktp/hba/ghd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/dnet/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/scsi/adapters/arcmsr/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/scsi/targets/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/vgatext/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/io/heci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/nskern/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/intel/os/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
249 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/pcbe/%.c
250         $(COMPILE.c) -o $@ $<
251         $(CTFCONVERT_O)

253 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/promif/%.c
254         $(COMPILE.c) -o $@ $<
255         $(CTFCONVERT_O)

257 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/syscall/%.c
258         $(COMPILE.c) -o $@ $<
259         $(CTFCONVERT_O)

261 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/os/%.c
262         $(COMPILE.c) -o $@ $<
263         $(CTFCONVERT_O)

265 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/kdi/%.c
266         $(COMPILE.c) -o $@ $<
267         $(CTFCONVERT_O)

269 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/kdi/%.s
270         $(COMPILE.s) -o $@ $<

272 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/kdi/$(SUBARCH_DIR)/%.s
273         $(COMPILE.s) -o $@ $<

275 $(OBJS_DIR)/%.o:                $(UTSBASE)/intel/zfs/%.c
276         $(COMPILE.c) -o $@ $<
277         $(CTFCONVERT_O)

279 #
280 # krtld compiled into unix
281 #

283 KRTLD_INC_PATH  = -I$(UTSBASE)/common/krtld -I$(UTSBASE)/intel/sys
284 KRTLD_INC_PATH  += -I$(UTSBASE)/intel/$(SUBARCH_DIR)/krtld

286 KRTLD_CPPFLAGS_32       = -DELF_TARGET_386
287 KRTLD_CPPFLAGS_64       = -DELF_TARGET_AMD64 -DMODDIR_SUFFIX=\"amd64\"
288 KRTLD_CPPFLAGS          = $(KRTLD_CPPFLAGS_$(CLASS)) -D_KRTLD

290 $(OBJS_DIR)/%.o:        $(UTSBASE)/common/krtld/%.c
291         $(COMPILE.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
292         $(CTFCONVERT_O)

294 $(OBJS_DIR)/%.o:        $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.c
295         $(COMPILE.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
296         $(CTFCONVERT_O)

298 #
299 # _DBOOT indicates that krtld is called from a dboot ELF section
300 #
301 $(OBJS_DIR)/kobj.o              := CPPFLAGS += -D_DBOOT

303 $(OBJS_DIR)/%.o:        $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.s
304         $(COMPILE.s) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
305         $(CTFCONVERT_O)

307 $(OBJS_DIR)/%.o:        $(SRC)/common/util/$(SUBARCH_DIR)/%.c
308         $(COMPILE.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
309         $(CTFCONVERT_O)


312 #
313 #       Section 1b: Lint ‘object’ build rules.
```



```
314 #
315 $(LINTS_DIR)/%.ln:              $(SRC)/common/fs/%.c
316         @($(LHEAD) $(LINT.c) $< $(LTAIL))

318 $(LINTS_DIR)/%.ln:              $(SRC)/common/util/i386/%.s
319         @($(LHEAD) $(LINT.s) $< $(LTAIL))

321 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/brand/sn1/%.s
322         @($(LHEAD) $(LINT.s) $< $(LTAIL))

324 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/brand/solaris10/%.s
325         @($(LHEAD) $(LINT.s) $< $(LTAIL))

327 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/dtrace/%.c
328         @($(LHEAD) $(LINT.c) $< $(LTAIL))

330 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/dtrace/%.s
331         @($(LHEAD) $(LINT.s) $< $(LTAIL))

333 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/zfs/%.c
334         @($(LHEAD) $(LINT.c) $< $(LTAIL))

336 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/fs/proc/%.c
337         @($(LHEAD) $(LINT.c) $< $(LTAIL))

339 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/ia32/ml/%.s
340         @($(LHEAD) $(LINT.s) $< $(LTAIL))

342 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/ia32/os/%.c
343         @($(LHEAD) $(LINT.c) $< $(LTAIL))

345 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/ia32/promif/%.c
346         @($(LHEAD) $(LINT.c) $< $(LTAIL))

348 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/ia32/syscall/%.c
349         @($(LHEAD) $(LINT.c) $< $(LTAIL))

351 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/%.c
352         @($(LHEAD) $(LINT.c) $< $(LTAIL))

354 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/%.s
355         @($(LHEAD) $(LINT.s) $< $(LTAIL))

357 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/%.c
358         @($(LHEAD) $(LINT.c) $< $(LTAIL))

360 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/debugger/%.c
358 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/debugger/%.c
361         @($(LHEAD) $(LINT.c) $< $(LTAIL))

363 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/events/%.c
361 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/events/%.c
364         @($(LHEAD) $(LINT.c) $< $(LTAIL))

366 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/hardware/%.c
364 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/hardware/%.c
367         @($(LHEAD) $(LINT.c) $< $(LTAIL))

369 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/dispatcher/%.c
367 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/dispatcher/%.c
370         @($(LHEAD) $(LINT.c) $< $(LTAIL))

372 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/executer/%.c
370 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/executer/%.c
373         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
375 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/parser/%.c
373 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/parser/%.c
376         @($(LHEAD) $(LINT.c) $< $(LTAIL))

378 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/namespace/%.c
376 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/namespace/%.c
379         @($(LHEAD) $(LINT.c) $< $(LTAIL))

381 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/resources/%.c
379 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/resources/%.c
382         @($(LHEAD) $(LINT.c) $< $(LTAIL))

384 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/tables/%.c
382 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/tables/%.c
385         @($(LHEAD) $(LINT.c) $< $(LTAIL))

387 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/utilities/%.c
385 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/utilities/%.c
388         @($(LHEAD) $(LINT.c) $< $(LTAIL))

390 $(LINTS_DIR)/%.ln:              $(SRC)/common/acpica/components/disassembler/%.c
388 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/acpica/disassembler/%.c
391         @($(LHEAD) $(LINT.c) $< $(LTAIL))

393 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/agpgart/%.c
394         @($(LHEAD) $(LINT.c) $< $(LTAIL))

396 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/agpmaster/%.c
397         @($(LHEAD) $(LINT.c) $< $(LTAIL))

399 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/amd8111s/%.c
400         @($(LHEAD) $(LINT.c) $< $(LTAIL))

402 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/amr/%.c
403         @($(LHEAD) $(LINT.c) $< $(LTAIL))

405 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/scsi/adapters/arcmsr/%.c
406         @($(LHEAD) $(LINT.c) $< $(LTAIL))

408 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/drm/%.c
409         @($(LHEAD) $(LINT.c) $< $(LTAIL))

411 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/hotplug/pcicfg/%.c
412         @($(LHEAD) $(LINT.c) $< $(LTAIL))

414 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/hotplug/pci/%.c
415         @($(LHEAD) $(LINT.c) $< $(LTAIL))

417 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/intel_nb5000/%.c
418         @($(LHEAD) $(LINT.c) $< $(LTAIL))

420 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/intel_nhm/%.c
421         @($(LHEAD) $(LINT.c) $< $(LTAIL))

423 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/ipmi/%.c
424         @($(LHEAD) $(LINT.c) $< $(LTAIL))

426 $(LINTS_DIR)/%.ln:              $(SRC)/common/mc/mc-amd/%.c
427         @($(LHEAD) $(LINT.c) $< $(LTAIL))

429 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/mc-amd/%.c
430         @($(LHEAD) $(LINT.c) $< $(LTAIL))

432 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/pci/%.c
433         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
435 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/pciex/%.c
436         @($(LHEAD) $(LINT.c) $< $(LTAIL))

438 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dktp/controller/ata/%.c
439         @($(LHEAD) $(LINT.c) $< $(LTAIL))

441 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dktp/dcdev/%.c
442         @($(LHEAD) $(LINT.c) $< $(LTAIL))

444 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dktp/disk/%.c
445         @($(LHEAD) $(LINT.c) $< $(LTAIL))

447 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dktp/drvobj/%.c
448         @($(LHEAD) $(LINT.c) $< $(LTAIL))

450 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dktp/hba/ghd/%.c
451         @($(LHEAD) $(LINT.c) $< $(LTAIL))

453 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/dnet/%.c
454         @($(LHEAD) $(LINT.c) $< $(LTAIL))

456 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/scsi/targets/%.c
457         @($(LHEAD) $(LINT.c) $< $(LTAIL))

459 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/vgatext/%.c
460         @($(LHEAD) $(LINT.c) $< $(LTAIL))

462 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/io/heci/%.c
463         @($(LHEAD) $(LINT.c) $< $(LTAIL))

465 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/os/%.c
466         @($(LHEAD) $(LINT.c) $< $(LTAIL))

468 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/pcbe/%.c
469         @($(LHEAD) $(LINT.c) $< $(LTAIL))

471 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/promif/%.c
472         @($(LHEAD) $(LINT.c) $< $(LTAIL))

474 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/syscall/%.c
475         @($(LHEAD) $(LINT.c) $< $(LTAIL))

477 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/os/%.c
478         @($(LHEAD) $(LINT.c) $< $(LTAIL))

480 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/kdi/%.c
481         @($(LHEAD) $(LINT.c) $< $(LTAIL))

483 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/kdi/%.s
484         @($(LHEAD) $(LINT.s) $< $(LTAIL))

486 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/kdi/$(SUBARCH_DIR)/%.s
487         @($(LHEAD) $(LINT.s) $< $(LTAIL))

489 $(LINTS_DIR)/%.ln:              $(UTSBASE)/intel/nskern/%.s
490         @($(LHEAD) $(LINT.s) $< $(LTAIL))

492 #
493 # krtld lints
494 #
495 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/krtld/%.c
496         @($(LHEAD) $(LINT.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))

498 $(LINTS_DIR)/%.ln:      $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.c
499         @($(LHEAD) $(LINT.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))
```

```
 501 $(LINTS_DIR)/%.ln:      $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.s
 502         @($(LHEAD) $(LINT.s) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))

 504 $(LINTS_DIR)/%.ln:      $(SRC)/common/util/$(SUBARCH_DIR)/%.c
 505         @($(LHEAD) $(LINT.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))

 507 $(OBJS_DIR)/kobj.ln             := CPPFLAGS += -D_DBOOT
```

```
**********************************************************
    1595 Thu Dec 26 13:50:37 2013
new/usr/src/uts/intel/acpica/Makefile
PANKOVs restructure
**********************************************************
```

```
   1 #
   2 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
   3 # Use is subject to license terms.
   4 #
   5 #
   6 #       This makefile drives the production of the ACPI CA services
   7 #       kernel module.
   8 #
   9 #       intel architecture dependent
  10 #

  12 #
  13 #       Path to the base of the uts directory tree (usually /usr/src/uts).
  14 #
  15 UTSBASE         = ../..

  17 #
  18 #       Define the module and object file sets.
  19 #
  20 MODULE          = acpica
  21 OBJECTS         = $(ACPICA_OBJS:%=$(OBJS_DIR)/%)
  22 LINTS           = $(ACPICA_OBJS:%.o=$(LINTS_DIR)/%.ln)
  23 ROOTMODULE      = $(ROOT_MISC_DIR)/$(MODULE)
  24 INC_PATH        += -I$(UTSBASE)/intel/sys/acpi
  25 INC_PATH        += -I$(UTSBASE)/i86pc
  26 INC_PATH        += -I$(SRC)/common
  27 INC_PATH        += -I$(SRC)/common/acpica/include

  29 #
  30 #       Include common rules.
  31 #
  32 include $(UTSBASE)/intel/Makefile.intel

  34 #
  35 #       Define targets
  36 #
  37 ALL_TARGET      = $(BINARY) $(CONFMOD)
  38 LINT_TARGET     = $(MODULE).lint
  39 INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

  41 #
  42 #       Overrides.
  43 #
  44 DEBUG_DEFS      += $(DEBUG_FLGS)

  46 #
  47 # lint pass one non-enforcement
  48 #
  49 CFLAGS          += $(CCVERBOSE) -DACPI_USE_LOCAL_CACHE
  48 CFLAGS += $(CCVERBOSE) -DPWRDMN -DACPI_USE_LOCAL_CACHE -DACPI_DEBUG_OUTPUT

  51 #
  52 # 3rd party code is not lint clean
  53 #
  53 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

  54 LINTFLAGS       += -errchk=%none
  55 LINTFLAGS       += -errhdr=%none
  56 LINTFLAGS       += -erroff=%all
  57 LINTFLAGS       += -errwarn=%none
```



```
  59 CERRWARN        += -_cc=-erroff=E_STATEMENT_NOT_REACHED

  61 CERRWARN        += -_gcc=-Wno-unused-variable
  62 CERRWARN        += -_gcc=-Wno-parentheses
  63 CERRWARN        += -_gcc=-Wno-uninitialized

  65 #
  66 #       Default build targets.
  67 #
  68 .KEEP_STATE:

  70 def:            $(DEF_DEPS)

  72 all:            $(ALL_DEPS)

  74 clean:          $(CLEAN_DEPS)

  76 clobber:        $(CLOBBER_DEPS)

  78 lint:           $(LINT_DEPS)

  80 modlintlib:     $(MODLINTLIB_DEPS)

  82 clean.lint:     $(CLEAN_LINT_DEPS)

  84 install:        $(INSTALL_DEPS)

  86 #
  87 #       Include common targets.
  88 #
  89 include $(UTSBASE)/intel/Makefile.targ
```

```
**********************************************************
   28791 Thu Dec 26 13:50:38 2013
new/usr/src/uts/intel/io/acpica/acpi_enum.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2012 Gary Mills
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * ACPI enumerator
 */

#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <sys/note.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <util/sscanf.h>

#include <acpica/include/acpi.h>
#include <sys/acpica.h>

static char keyboard_alias[] = "keyboard";
static char mouse_alias[] = "mouse";
#define ACPI_ENUM_DEBUG         "acpi_enum_debug"
#define PARSE_RESOURCES_DEBUG   0x0001
#define MASTER_LOOKUP_DEBUG     0x0002
#define DEVICES_NOT_ENUMED      0x0004
#define PARSE_RES_IRQ           0x0008
#define PARSE_RES_DMA           0x0010
#define PARSE_RES_MEMORY        0x0020
#define PARSE_RES_IO            0x0040
#define PARSE_RES_ADDRESS       0x0080
#define ISA_DEVICE_ENUM         0x1000
#define PROCESS_CIDS            0x2000
static unsigned long acpi_enum_debug = 0x00;

static char USED_RESOURCES[] = "used-resources";
```



```
static dev_info_t *usedrdip = NULL;
static unsigned short used_interrupts = 0;
static unsigned short used_dmas = 0;
typedef struct used_io_mem {
        unsigned int start_addr;
        unsigned int length;
        struct used_io_mem *next;
} used_io_mem_t;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   22335 Thu Dec 26 13:50:38 2013
new/usr/src/uts/intel/io/acpica/acpica.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2011, Joyent, Inc. All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * Copyright (c) 2009, Intel Corporation.
 * All rights reserved.
 */

/*
 * x86 ACPI CA services
 * Solaris x86 ACPI CA services
 */

#include <sys/file.h>
#include <sys/errno.h>
#include <sys/conf.h>
#include <sys/modctl.h>
#include <sys/open.h>
#include <sys/stat.h>
#include <sys/spl.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/esunddi.h>
#include <sys/kstat.h>
#include <sys/x86_archext.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/archsystm.h>

/*
 *
 */
static  struct modlmisc modlmisc = {
        &mod_miscops,
        "ACPI interpreter",
};
```



```
_____unchanged_portion_omitted_


/*
 * Process acpi-user-options property  if present
 */
static void
acpica_process_user_options()
{
        static int processed = 0;
        int acpi_user_options;
        char *acpi_prop;

        /*
         * return if acpi-user-options has already been processed
         */
        if (processed)
                return;
        else
                processed = 1;

        /* converts acpi-user-options from type string to int, if any */
        if (ddi_prop_lookup_string(DDI_DEV_T_ANY, ddi_root_node(),
            DDI_PROP_DONTPASS, "acpi-user-options", &acpi_prop) ==
            DDI_PROP_SUCCESS) {
                long data;
                int ret;
                ret = ddi_strtol(acpi_prop, NULL, 0, &data);
                if (ret == 0) {
                        e_ddi_prop_remove(DDI_DEV_T_NONE, ddi_root_node(),
                            "acpi-user-options");
                        e_ddi_prop_update_int(DDI_DEV_T_NONE, ddi_root_node(),
                            "acpi-user-options", data);
                }
                ddi_prop_free(acpi_prop);
        }

        /*
         * fetch the optional options property
         */
        acpi_user_options = ddi_prop_get_int(DDI_DEV_T_ANY, ddi_root_node(),
            DDI_PROP_DONTPASS, "acpi-user-options", 0);

        /*
         * Note that 'off' has precedence over 'on'
         * Also note - all cases of ACPI_OUSER_MASK
         * provided here, no default: case is present
         */
        switch (acpi_user_options & ACPI_OUSER_MASK) {
        case ACPI_OUSER_DFLT:
                acpica_enable = acpica_check_bios_date(1999, 1, 1);
                break;
        case ACPI_OUSER_ON:
                acpica_enable = TRUE;
                break;
        case ACPI_OUSER_OFF:
        case ACPI_OUSER_OFF | ACPI_OUSER_ON:
                acpica_enable = FALSE;
                break;
        }

        acpi_init_level = ACPI_FULL_INITIALIZATION;

        /*
```

```
388          * special test here; may be generalized in the
389          * future - test for a machines that are known to
390          * work only in legacy mode, and set OUSER_LEGACY if
391          * we're on one
392          */
393         if (acpica_metro_old_bios())
394                 acpi_user_options |= ACPI_OUSER_LEGACY;

396         /*
397          * If legacy mode is specified, set initialization
398          * options to avoid entering ACPI mode and hooking SCI
399          * - basically try to act like legacy acpi_intp
400          */
401         if ((acpi_user_options & ACPI_OUSER_LEGACY) != 0)
402                 acpi_init_level |= (ACPI_NO_ACPI_ENABLE | ACPI_NO_HANDLER_INIT);

404         /*
405          * modify default ACPI CA debug output level for non-DEBUG builds
406          * (to avoid BIOS debug chatter in /var/adm/messages)
407          */
408         if (acpica_muzzle_debug_output) {
409                 AcpiDbgLayer = 0;
405         if (acpica_muzzle_debug_output)
410                 AcpiDbgLevel = 0;
411         }
412 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   16042 Thu Dec 26 13:50:39 2013
new/usr/src/uts/intel/io/acpica/acpica_ec.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2011 Joyent, Inc.  All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * x86 ACPI CA Embedded Controller operation region handler
 * Solaris x86 ACPI CA Embedded Controller operation region handler
 */

#include <sys/file.h>
#include <sys/errno.h>
#include <sys/conf.h>
#include <sys/modctl.h>
#include <sys/open.h>
#include <sys/stat.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/note.h>
#include <sys/atomic.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

/*
 * EC status bits
 * Low to high
 *      Output buffer full?
 *      Input buffer full?
 *      <reserved>
 *      Data register is command byte?
 *      Burst mode enabled?
 *      SCI event?
 *      SMI event?
 *      <reserved>
 */
```



```
#define EC_OBF  (0x01)
#define EC_IBF  (0x02)
#define EC_DRC  (0x08)
#define EC_BME  (0x10)
#define EC_SCI  (0x20)
#define EC_SMI  (0x40)

/*
 * EC commands
 */
#define EC_RD   (0x80)
#define EC_WR   (0x81)
#define EC_BE   (0x82)
#define EC_BD   (0x83)
#define EC_QR   (0x84)

#define IO_PORT_DES (0x47)

/*
 * EC softstate
 */
static struct ec_softstate {
        uint8_t  ec_ok;         /* != 0 if we have ec_base, ec_sc */
        uint16_t ec_base;       /* base of EC I/O port - data */
        uint16_t ec_sc;         /* EC status/command */
        ACPI_HANDLE ec_dev_hdl; /* EC device handle */
        ACPI_HANDLE ec_gpe_hdl; /* GPE info */
        ACPI_INTEGER ec_gpe_bit;
        kmutex_t    ec_mutex;   /* serialize access to EC */
} ec;
______unchanged_portion_omitted_
```

****
8653 Thu Dec 26 13:50:41 2013
new/usr/src/uts/intel/io/acpica/master_ops.c
PANKOVs restructure
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

#include <sys/kobj.h>
#include <sys/kobj_lex.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

#define masterfile "/boot/solaris/devicedb/master"

/*
 * Internal definitions
 */

typedef enum {
        MF_UNEXPECTED = -1,
        MF_IDENT,
        MF_STRING,
        MF_EOF,
        MF_NEWLINE,
        MF_EQUALS,
        MF_BIT_OR
} mftoken_t;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   51515 Thu Dec 26 13:50:42 2013
new/usr/src/uts/intel/io/acpica/osl.c
acpica-unix2-20130823
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2012 Joyent, Inc.  All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 PALO, Richard. All rights reserved.
 * Copyright 2011 Joyent, Inc.  All rights reserved.
 */

/*
 * Copyright (c) 2009-2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * x86 ACPI CA OSL
 * ACPI CA OSL for Solaris x86
 */

#include <sys/types.h>
#include <sys/kmem.h>
#include <sys/psm.h>
#include <sys/pci_cfgspace.h>
#include <sys/apic.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <sys/pci.h>
#include <sys/kobj.h>
#include <sys/taskq.h>
#include <sys/strlog.h>
#include <sys/x86_archext.h>
#include <sys/note.h>
#include <sys/promif.h>

#include <acpica/include/accommon.h>
#include <sys/acpi/accommon.h>
#include <sys/acpica.h>
```



```
#define MAX_DAT_FILE_SIZE       (64*1024)

/* local functions */
static int CompressEisaID(char *np);

static void scan_d2a_subtree(dev_info_t *dip, ACPI_HANDLE acpiobj, int bus);
static int acpica_query_bbn_problem(void);
static int acpica_find_pcibus(int busno, ACPI_HANDLE *rh);
static int acpica_eval_hid(ACPI_HANDLE dev, char *method, int *rint);
static ACPI_STATUS acpica_set_devinfo(ACPI_HANDLE, dev_info_t *);
static ACPI_STATUS acpica_unset_devinfo(ACPI_HANDLE);
static void acpica_devinfo_handler(ACPI_HANDLE, void *);

/*
 * Event queue vars
 */
int acpica_eventq_init = 0;
ddi_taskq_t *osl_eventq[OSL_EC_BURST_HANDLER+1];

/*
 * Priorities relative to minclsyspri that each taskq
 * run at; OSL_NOTIFY_HANDLER needs to run at a higher
 * priority than OSL_GPE_HANDLER.  There's an implicit
 * assumption that no priority here results in exceeding
 * maxclsyspri.
 * Note: these initializations need to match the order of
 * ACPI_EXECUTE_TYPE.
 */
int osl_eventq_pri_delta[OSL_EC_BURST_HANDLER+1] = {
        0,      /* OSL_GLOBAL_LOCK_HANDLER */
        2,      /* OSL_NOTIFY_HANDLER */
        0,      /* OSL_GPE_HANDLER */
        0,      /* OSL_DEBUGGER_THREAD */
        0,      /* OSL_EC_POLL_HANDLER */
        0       /* OSL_EC_BURST_HANDLER */
};
______unchanged_portion_omitted_

ACPI_STATUS
AcpiOsPhysicalTableOverride(ACPI_TABLE_HEADER *ExistingTable,
    ACPI_PHYSICAL_ADDRESS *NewAddress, UINT32 *NewTableLength)
{
        return (AE_SUPPORT);
}

/*
 * ACPI semaphore implementation
 */
typedef struct {
        kmutex_t        mutex;
        kcondvar_t      cv;
        uint32_t        available;
        uint32_t        initial;
        uint32_t        maximum;
} acpi_sema_t;
______unchanged_portion_omitted_

void
AcpiOsWaitEventsComplete (void)
{
        if (acpica_eventq_init) {
                int     i;
                /*
                 * blocks until all events initiated by AcpiOsExecute have compl
                 */
                for (i = OSL_GLOBAL_LOCK_HANDLER; i <= OSL_EC_BURST_HANDLER; i++
```

```
                        if (osl_eventq[i])
                                ddi_taskq_wait(osl_eventq[i]);
                }
        }
    return;
}

void
AcpiOsSleep(ACPI_INTEGER Milliseconds)
{
        /*
         * During kernel startup, before the first tick interrupt
         * has taken place, we can't call delay; very late in
         * kernel shutdown or suspend/resume, clock interrupts
         * are blocked, so delay doesn't work then either.
         * So we busy wait if lbolt == 0 (kernel startup)
         * or if acpica_use_safe_delay has been set to a
         * non-zero value.
         */
        if ((ddi_get_lbolt() == 0) || acpica_use_safe_delay)
                drv_usecwait(Milliseconds * 1000);
        else
                delay(drv_usectohz(Milliseconds * 1000));
}
```

*unchanged_portion_omitted_*

```


/*
 *
 */

#define OSL_RW(ptr, val, type, rw) \
        { if (rw) *((type *)(ptr)) = *((type *) val); \
            else *((type *) val) = *((type *)(ptr)); }


static void
osl_rw_memory(ACPI_PHYSICAL_ADDRESS Address, UINT64 *Value,
osl_rw_memory(ACPI_PHYSICAL_ADDRESS Address, UINT32 *Value,
    UINT32 Width, int write)
{
        size_t  maplen = Width / 8;
        caddr_t ptr;

        ptr = psm_map_new((paddr_t)Address, maplen,
            PSM_PROT_WRITE | PSM_PROT_READ);

        switch (maplen) {
        case 1:
                OSL_RW(ptr, Value, uint8_t, write);
                break;
        case 2:
                OSL_RW(ptr, Value, uint16_t, write);
                break;
        case 4:
                OSL_RW(ptr, Value, uint32_t, write);
                break;
        case 8:
                OSL_RW(ptr, Value, uint64_t, write);
                break;
        default:
                cmn_err(CE_WARN, "!osl_rw_memory: invalid size %d",
                    Width);
                break;
        }
```



```
        psm_unmap(ptr, maplen);
}

ACPI_STATUS
AcpiOsReadMemory(ACPI_PHYSICAL_ADDRESS Address,
                UINT64 *Value, UINT32 Width)
                UINT32 *Value, UINT32 Width)
{
        osl_rw_memory(Address, Value, Width, 0);
        return (AE_OK);
}

ACPI_STATUS
AcpiOsWriteMemory(ACPI_PHYSICAL_ADDRESS Address,
                UINT64 Value, UINT32 Width)
                UINT32 Value, UINT32 Width)
{
        osl_rw_memory(Address, &Value, Width, 1);
        return (AE_OK);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
   96795 Thu Dec 26 13:50:44 2013
new/usr/src/uts/intel/io/pci/pci_boot.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <sys/sunndi.h>
#include <sys/pci.h>
#include <sys/pci_impl.h>
#include <sys/pcie_impl.h>
#include <sys/memlist.h>
#include <sys/bootconf.h>
#include <io/pci/mps_table.h>
#include <sys/pci_cfgacc.h>
#include <sys/pci_cfgspace.h>
#include <sys/pci_cfgspace_impl.h>
#include <sys/psw.h>
#include "../../../../common/pci/pci_strings.h"
#include <sys/apic.h>
#include <io/pciex/pcie_nvidia.h>
#include <sys/hotplug/pci/pciehpc_acpi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/iommulib.h>
#include <sys/devcache.h>
#include <sys/pci_cfgacc_x86.h>

#define pci_getb        (*pci_getb_func)
#define pci_getw        (*pci_getw_func)
#define pci_getl        (*pci_getl_func)
#define pci_putb        (*pci_putb_func)
#define pci_putw        (*pci_putw_func)
#define pci_putl        (*pci_putl_func)
#define dcmn_err        if (pci_boot_debug) cmn_err

#define CONFIG_INFO     0
#define CONFIG_UPDATE   1
#define CONFIG_NEW      2
#define CONFIG_FIX      3
```



```
#define COMPAT_BUFSIZE  512

#define PPB_IO_ALIGNMENT        0x1000          /* 4K aligned */
#define PPB_MEM_ALIGNMENT       0x100000        /* 1M aligned */
/* round down to nearest power of two */
#define P2LE(align)                                             \
        {                                                       \
                int i = 0;                                      \
                while (align >>= 1)                             \
                        i ++;                                   \
                align = 1 << i;                                 \
        }

/* for is_vga and list_is_vga_only */

enum io_mem {
        IO,
        MEM
};
______unchanged_portion_omitted_
```

```
**********************************************************
   16007 Thu Dec 26 13:50:45 2013
new/usr/src/uts/intel/io/pci/pci_resource.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * pci_resource.c -- routines to retrieve available bus resources from
 *                the MP Spec. Table and Hotplug Resource Table
 */

#include <sys/types.h>
#include <sys/memlist.h>
#include <sys/pci_impl.h>
#include <sys/systm.h>
#include <sys/cmn_err.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include "mps_table.h"
#include "pcihrt.h"

extern int pci_boot_debug;
extern int pci_bios_maxbus;
#define dprintf if (pci_boot_debug) printf

static int tbl_init = 0;
static uchar_t *mps_extp = NULL;
static uchar_t *mps_ext_endp = NULL;
static struct php_entry *hrt_hpep;
static int hrt_entry_cnt = 0;
static int acpi_cb_cnt = 0;

static void mps_probe(void);
static void acpi_pci_probe(void);
static int mps_find_bus_res(int, int, struct memlist **);
static void hrt_probe(void);
static int hrt_find_bus_res(int, int, struct memlist **);
static int acpi_find_bus_res(int, int, struct memlist **);
static uchar_t *find_sig(uchar_t *cp, int len, char *sig);
static int checksum(unsigned char *cp, int len);
static ACPI_STATUS acpi_wr_cb(ACPI_RESOURCE *rp, void *context);
void bus_res_fini(void);
static void acpi_trim_bus_ranges(void);
```



```
struct memlist *acpi_io_res[256];
struct memlist *acpi_mem_res[256];
struct memlist *acpi_pmem_res[256];
struct memlist *acpi_bus_res[256];

/*
 * -1 = attempt ACPI resource discovery
 *  0 = don't attempt ACPI resource discovery
 *  1 = ACPI resource discovery successful
 */
volatile int acpi_resource_discovery = -1;

struct memlist *
find_bus_res(int bus, int type)
{
        struct memlist *res = NULL;

        if (tbl_init == 0) {
                tbl_init = 1;
                acpi_pci_probe();
                hrt_probe();
                mps_probe();
        }

        if (acpi_find_bus_res(bus, type, &res) > 0)
                return (res);

        if (hrt_find_bus_res(bus, type, &res) > 0)
                return (res);

        (void) mps_find_bus_res(bus, type, &res);
        return (res);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    7920 Thu Dec 26 13:50:45 2013
new/usr/src/uts/intel/io/pciex/pcie_acpi.c
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
#include <sys/sysmacros.h>
#include <sys/types.h>
#include <sys/kmem.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <sys/promif.h>
#include <sys/pcie.h>
#include <sys/pci_cap.h>
#include <sys/pcie_impl.h>
#include <sys/pcie_acpi.h>
#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>

ACPI_STATUS pcie_acpi_eval_osc(dev_info_t *dip, ACPI_HANDLE osc_hdl,
        uint32_t *osc_flags);
static ACPI_STATUS pcie_acpi_find_osc(ACPI_HANDLE busobj,
        ACPI_HANDLE *osc_hdlp);

#ifdef DEBUG
static void pcie_dump_acpi_obj(ACPI_HANDLE pcibus_obj);
static ACPI_STATUS pcie_walk_obj_namespace(ACPI_HANDLE hdl, uint32_t nl,
        void *context, void **ret);
static ACPI_STATUS pcie_print_acpi_name(ACPI_HANDLE hdl, uint32_t nl,
        void *context, void **ret);
#endif /* DEBUG */

int
pcie_acpi_osc(dev_info_t *dip, uint32_t *osc_flags)
{
        ACPI_HANDLE pcibus_obj;
        int status = AE_ERROR;
        ACPI_HANDLE osc_hdl;
        pcie_bus_t *bus_p = PCIE_DIP2BUS(dip);
        pcie_x86_priv_t *osc_p = (pcie_x86_priv_t *)bus_p->bus_plat_private;
```



```
        /* Mark this so we know _OSC has been called for this device */
        osc_p->bus_osc = B_TRUE;

        /*
         * (1)  Find the ACPI device node for this bus node.
         */
        status = acpica_get_handle(dip, &pcibus_obj);
        if (status != AE_OK) {
                PCIE_DBG("No ACPI device found (dip %p)\n", (void *)dip);
                return (DDI_FAILURE);
        }

        /*
         * (2)  Check if _OSC method is present.
         */
        if (pcie_acpi_find_osc(pcibus_obj, &osc_hdl) != AE_OK) {
                /* no _OSC method present */
                PCIE_DBG("no _OSC method present for dip %p\n",
                    (void *)dip);
                return (DDI_FAILURE);
        }

        /*
         * (3)  _OSC method exists; evaluate _OSC.
         */
        if (pcie_acpi_eval_osc(dip, osc_hdl, osc_flags) != AE_OK) {
                PCIE_DBG("Failed to evaluate _OSC method for dip 0x%p\n",
                    (void *)dip);
                return (DDI_FAILURE);
        }

        osc_p->bus_osc_hp = (*osc_flags & OSC_CONTROL_PCIE_NAT_HP) ?
            B_TRUE : B_FALSE;
        osc_p->bus_osc_aer = (*osc_flags & OSC_CONTROL_PCIE_ADV_ERR) ?
            B_TRUE : B_FALSE;

#ifdef DEBUG
        if (pcie_debug_flags > 1)
                pcie_dump_acpi_obj(pcibus_obj);
#endif /* DEBUG */

        return (DDI_SUCCESS);
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
    2322 Thu Dec 26 13:50:46 2013
new/usr/src/uts/intel/pci_autoconfig/Makefile
PANKOVs restructure
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# uts/intel/pci_autoconfig/Makefile
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#       This makefile drives the production of the PCI autoconfiguration
#       kernel module.
#
#       intel platform dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = pci_autoconfig
OBJECTS         = $(PCI_AUTOCONFIG_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(PCI_AUTOCONFIG_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_MISC_DIR)/$(MODULE)
INC_PATH        += -I$(UTSBASE)/i86pc
INC_PATH        += -I$(SRC)/common

#
#       Include common rules.
#
include $(UTSBASE)/intel/Makefile.intel

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

#
# Depends on acpica ACPI CA interpreter and PCI-E framework
#
```



```
LDFLAGS         += -dy -Nmisc/acpica -Nmisc/pcie

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

CERRWARN        += -_gcc=-Wno-parentheses

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

#
#       Include common targets.
#
include $(UTSBASE)/intel/Makefile.targ
```

```
**********************************************************
    2365 Thu Dec 26 13:50:46 2013
new/usr/src/uts/intel/power/Makefile
PANKOVs restructure
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# uts/intel/power/Makefile
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
#       This makefile drives the production of the power driver
#
#       intel architecture dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = power
OBJECTS         = $(POWER_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(POWER_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_DRV_DIR)/$(MODULE)
CONF_SRCDIR     = $(UTSBASE)/common/io
INC_PATH        += -I$(SRC)/common

#
#       Include common rules.
#
include $(UTSBASE)/intel/Makefile.intel

#
#       Define targets
#
ALL_TARGET      = $(BINARY) $(CONFMOD)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE) $(ROOT_CONFFILE)

#
# lint pass one enforcement
```



```
#
CFLAGS += $(CCVERBOSE)

#
# ACPI power button
#
CFLAGS += -DACPI_POWER_BUTTON
LINTFLAGS += -DACPI_POWER_BUTTON

LDFLAGS += -dy -N misc/acpica

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_STATIC_UNUSED

CERRWARN        += -_gcc=-Wno-unused-variable

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

#
#       Include common targets.
#
include $(UTSBASE)/intel/Makefile.targ
```

```
**********************************************************
    2492 Thu Dec 26 13:50:48 2013
new/usr/src/uts/intel/sys/hotplug/pci/pciehpc_acpi.h
PANKOVs restructure
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _PCIEHPC_ACPI_H
#define _PCIEHPC_ACPI_H

#ifdef  __cplusplus
extern "C" {
#endif

#include <acpica/include/acpi.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/hotplug/pci/pcie_hp.h>
#include <sys/hotplug/pci/pciehpc.h>

/* soft state data structure for ACPI hot plug mode */
typedef struct pciehpc_acpi {
        /* handle for the ACPI device for the bus node with HPC */
        ACPI_HANDLE     bus_obj;

        /* handle for the ACPI device for the slot (dev#0,func#0) */
        ACPI_HANDLE     slot_dev_obj;

        /* ACPI control methods present on the bus node */
        uint16_t        bus_methods;

        /* ACPI control methods on the slot device functions */
        uint16_t        slot_methods;
} pciehpc_acpi_t;
```

*_____unchanged_portion_omitted_*